

Inside Unix

SOFTWARE TOOLS FOR ADVANCED PROGRAMMERS

Dr. Dobb's Journal

1988 December 1984

\$2.95 (3.50 Canada)

A File Browser

Unix:
Varieties,
Device Drivers
and Internals

Introduction
to Parsing



the closest thing to
perfect is WordPerfect
by SSI.

Reference Magazine

When it comes to software, nobody's perfect. But according to many of the experts, one word processing program is as close as you can get. No wonder it's called WordPerfect.

What are all the critics raving about?

Simplicity. Most WordPerfect functions require only one keystroke, a simple press of a finger. So you can concentrate on writing, not programming.

Speed. Because it is document-

oriented instead of page-oriented, WordPerfect won't make you

WordPerfect isn't flawless word processing software, but it comes very close.

Digital Review

wait between pages. No matter how fast you type, WordPerfect won't slow you down.

Features. From writers to doctors, accountants to lawyers, WordPerfect has built-in special functions to meet a wide variety of specific needs. And at SSI, every day is spent upgrading and improving WordPerfect — reaching for perfection.

Get your hands on the critics' choice, WordPerfect word processing from SSI.

WordPerfect is my favorite because it is easy, simple and powerful. The people

List Magazine

It's the closest thing to perfection.

For more information, see your dealer.

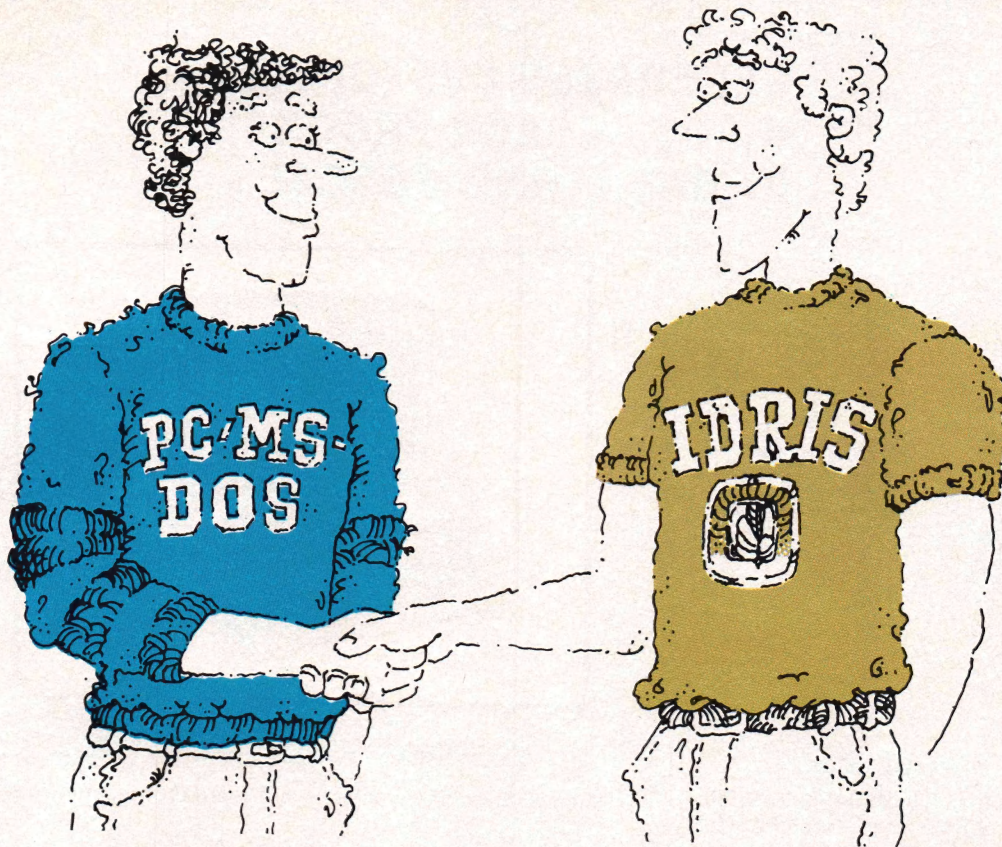
Or call or write:

SSI Software
288 West Center Street
Orem, Utah 84057
Information: (801) 224-4000
Order Desk: 1-800-321-4566,
Toll-free



SSI Software
Reaching for perfection.

The best of both worlds



Now, software developers can expand their markets and increase their productivity with Co-Idris™, the newest UNIX-like operating system from Whitesmiths, Ltd.

Co-Idris is a professional, sophisticated tool enabling users to develop programs in a powerful and flexible UNIX-like environment, then easily port these applications to a wide range of PC/MS-DOS machines, including the IBM PC, DEC Rainbow, Wang PC, DG Desktop, and Olympia PC. With the Co-Idris package, you can construct C, Pascal, or assembler programs for operation under Co-Idris, DOS or CP/M-86.

Able to work in as little as 128 KB of total main memory, Co-Idris allows **concurrent** access to both Idris-based programs and PC- or MS-DOS based application programs. You get the multi-user, multi-tasking features of a UNIX environment as well as the rich selection of DOS applications. And there is no need to reboot DOS, ever.

Co-Idris works on most all PC/MS-DOS based configurations with hard disks, and it's **available now!**

Dealer inquiries invited.



Whitesmiths, Ltd.

97 Lowell Road Concord, MA 01742 (617) 369-8499
TLX 750246 SOFTWARE CNCM

DISTRIBUTORS: Australia, Fawnray Pty. Ltd., Hurstville, (612) 570-6100; France Cosmic s.a.n.l. 76 Quai Des Carrieres, 94227 Charenton Le Pont 1-378-8357; Japan Advanced Data Controls Corp., Chiyoda-ku, Tokyo (03) 263-0383; United Kingdom, Real Time Systems, Douglas, Isle of Man 0624-26021; Sweden, Unisoft A.B., Goteborg, 31-125810. Rainbow is a trademark of Digital Equipment Corp. UNIX is a trademark of Bell Laboratories; MS-DOS is a trademark of Microsoft Corp. PC-DOS is a trademark of International Business Machines Corporation. Idris is a trademark of Whitesmiths, Ltd.

SCREEN SCULPTOR™

**GENERATES CUSTOMIZED
INPUT SCREEN PROGRAMS
IN BASIC AND PASCAL . . .**



THE
SOFTWARE
BOTTLING
COMPANY
OF NEW YORK



BASIC or PASCAL. Easily!

Generate programs in **BASIC** or **PASCAL**. Your choice. Works with * **Turbo PASCAL**, * **IBM PASCAL** or **IBM BASIC** (Interpreter and Compiler). **Easy** to use. Begin productive use in minutes!

POWERFUL SCREENS

Everyone can have **professional quality screens** to dress up any program. Generate **complex, colorful, effective** screens in minutes.

FULL FUNCTION SCREEN CREATION

Simply "draw" input screens with word processor style editor.

Advance screen creating features include:

- **Draw boxes, lines, etc.** in seconds with unique character selection menu.
- **Repeat last character** in any direction.
- Special **color-select screen** displays all available colors.
- **Paint and Repaint** sections of screen at any time.
- **Copy and Move** sections of screen.
- **Insert or Delete characters and lines.**
- **Input field definition screen** gives you **total control** of character type definitions, edit screen masks, input sequence, variable names, initial values, protected characters, etc.

COMPLETE DATA ENTRY ROUTINES

Generates customized program code that allows **professional quality** data input using the full PC keyboard (cursor keys, delete, insert, and more). Checks input data for valid entries and displays error messages.

Programs are easily merged with your own programs.

Easy to follow documentation shows how and where you can modify the generated programs.

- Available now with **IBM PC, PCjr, PCXT, and all true compatibles.**
- Requires **128k RAM, one floppy disk drive, and PC DOS.** Works with any 80 column display type.

* Turbo PASCAL is a registered trademark of Boreland International, Ltd. IBM is a registered trademark of IBM corporation.

**TO ORDER SEE YOUR LOCAL DEALER
OR CALL (718) 728-2200**

Only **\$125⁰⁰**

(Includes shipping and handling)
(N.Y.S. Res. Add 8 1/4 % sales tax)

Credit Card Orders call 1-800-824-7888. Operator 268

Alaska and Hawaii call 1-800-824-7919. Operator 268

DEALER INQUIRIES INVITED. SORRY, NO C.O.D.

Produced and Distributed by **The Software Bottling Co. of New York.**

29-14 23 Ave., New York City, N.Y. 11105

ITEM No. 1100

"Despite the recent press notices, multiuser microcomputers aren't anything new!"

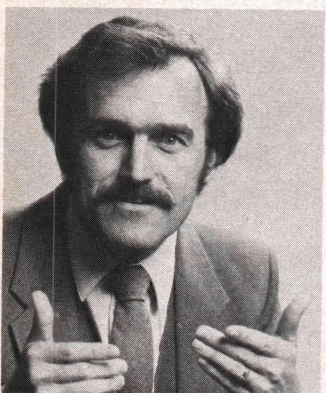
This is the first in a series of discussions with Rod Coleman, President of Stride Micro (formerly Sage Computer) on the 68000 multiuser market and its current environment.

Q: Why do you say that?

RC: "The technology to build a high performance multiuser system has been around for five years. And while some of the leaders in this industry have been pretending that micro multiuser didn't exist, we've been shipping complete systems for nearly three years. The benefits of multiuser are undeniable; it is more cost effective, and offers greater flexibility and utility. But until just recently, the marketing pressure to be compatible instead of being better, has blinded the industry."

Q: What do you mean?

RC: "Well, for example, the Motorola 68000 processor introduced 16/32-bit technology to the personal computer world a long time ago. It was fully capable of



"A surprising feature is compatibility. Everybody talks about it, but nobody does anything about it."

meeting high performance and multiuser design requirements in 1980. Instead of this trend taking off, most energy was spent promoting 8088/8086 products that

were clearly inferior from a technical point of view. This phenomenon leads me to believe that they will soon rewrite the old proverb: 'Build a better mousetrap and the world will beat a path to your door,' but only if they can find the way through the marketing fog."

Q: Are things changing now?

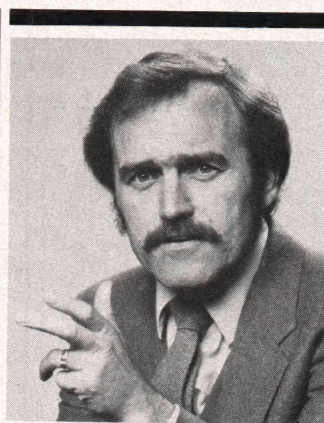
RC: "Yes and no. With the business world starting to take more and more interest in microcomputer solutions, the advantages of a solid multiuser system couldn't be kept hidden forever; companies like ours and a few others were beginning to make a dent. Instead of taking a fresh approach, some of the newest multiuser offerings will probably only give the technology an undeserved black eye! Multiuser is far more than the ability to plug in more terminals. It involves things like machine compatibility, fast processors, adequate memory, large storage capacities, backup features, networking, and operating system flexibility."

Q: Is this what makes the new Stride 400 Series different?

RC: "Exactly. That sounds self-serving, but it's true. Today a number of companies are introducing their first multiuser system. We've been building and shipping multiuser machines for almost three years. We know the pitfalls, we've fallen into some of them. But we have learned from our mistakes."

Q: Give me some examples.

RC: A hard disk is almost mandatory for any large multiuser installation. Yet, backing up a hard disk can be a nightmare if you only have floppies to work with. That's why we've added a tape backup option to all the larger Stride 400 Series machines. It's irresponsible for a manufacturer to market a multiuser system without such backup. Another good lesson was bus design. We started with one of our own designs, but learned that it's important not only to find a bus that is powerful, but also one that has good support and a strong future to serve tomorrow's needs. We



"The marketing pressure to be compatible instead of being better, has blinded the industry."

think the VMEbus is the only design that meets both criteria and thus have made it a standard feature of every Stride 400 Series machine."

Q: What are some of the other unique features of the 400 Series?

RC: "A surprising feature is compatibility. Everybody talks about it, but nobody does anything about it. Our systems are completely compatible with each other from the 420 model starting at \$2900, through the 440, on to the powerful 460 which tops out near \$60,000. Each system can talk to the others via the standard built-in local area network. Go ahead and compare this with others in the industry. You'll find their little machines don't talk to their big ones, or that the networking and multiuser are incompatible, or that they have different processors or operating systems, and so on."

Q: When you were still known as Sage Computer, you had a reputation for performance, is that still the case with the new Stride 400 Series?

RC: "Certainly, that's our calling card: 'Performance By Design.' Our new systems are actually faster; our standard processor is a 10 MHz 68000 running with no wait

states. That gives us a 25% increase over the Sage models. And, we have a 12 MHz processor as an option. Let me add that speed isn't the only way to judge performance. I think it is also measured in our flexibility. We support a dozen different operating systems, not just one. And our systems service a wide variety of applications from the garage software developer to the corporate consumer running high volume business applications."

Q: Isn't that the same thing all manufacturers say in their ads?

RC: "Sure it is. But to use another over used-term, 'shop around'. We like to think of our systems as 'full service 68000 supermicrocomputers.' Take a look at everyone else's literature and then compare. When you examine cost, performance, flexibility, and utility, we don't think there's anyone else in the race. Maybe that's why we've shipped and installed more multiuser 68000 systems than anyone else."



STRIDE
MICRO

Formerly Sage Computer

For more information on Stride or the location of the nearest Stride Dealer call or write us today. We'll also send you a free copy of our 32 page product catalog.

Corporate Offices:
4905 Energy Way
Reno, NV 89502
(702) 322-6868

Regional Offices:
Boston: (617) 229-6868
Dallas: (214) 392-7070

Dr. Dobb's Journal

Editorial

Editor-in-Chief Michael Swaine
Editor Reynold Wiggins
Managing Editor Randy Sutherland
Contributing Editors Robert Blum,
Dave Cortesi,
Ray Duncan,
Anthony Skjellum,
Michael Wiesenber
Copy Editors Polly Koch, Cindy Martin
Typesetter Jean Aring
Editorial Intern Mark Johnson

Production

Design/Production
Director Detta Penna
Art Director Shelley Rae Doeden
Production Assistant Alida Hinton
Cover Scott Kim

Advertising

Advertising Director Stephen Friedman
Advertising Sales Walter Andrzejewski,
Shawn Horst
Advertising Coordinators Alison Milne,
Lisa Boudreau
Advertising Sales Beth Dudas

Circulation

Circulation and
Promotions Director Beatrice Blatteis
Fulfillment Manager Stephanie Barber
Direct Response
Coordinator Maureen Snee
Promotions Coordinator Jane Sharninghouse
Single Copy Sales
Coordinator Sally Brenton
Single Copy Sales Lorraine McLaughlin
Circulation Assistant Kathleen Boyd

M&T Publishing, Inc.

Chairman of the Board Otmar Weber
Director C.F. von Quadt
President Laird Foshay

Entire contents copyright © 1984 by M&T Publishing, Inc. unless otherwise noted on specific articles. All rights reserved.

Dr. Dobb's Journal (USPS 307690) is published monthly by M&T Publishing, Inc., 2464 Embarcadero Way, Palo Alto, CA 94303, (415) 424-0600. Second class postage paid at Palo Alto and at additional entry points.

Address correction requested. Postmaster: Send Form 3579 to *Dr. Dobb's Journal*, 2464 Embarcadero Way, Palo Alto, CA 94303. **ISSN 0278-6508**

Subscription Rates: \$25 per year within the United States, \$44 for first class to Canada and Mexico, \$62 for airmail to other countries. Payment must be in U.S. Dollars, drawn on a U.S. Bank.

Contributing Subscribers: Christine Bell, W.D. Rausch, DeWitt S. Brown, Burks A. Smith, Robert C. Luckey, Transdata Corp., Mark Ketter, Friden Mailing Equipment, Frank Lawyer, Rodney Black, Kenneth Drexler, Real Paquin, Ed Malin, John Saylor Jr., Ted A. Reuss III, InfoWorld, Stan Veit, Western Material Control, S.P. Kennedy, John Hatch, Richard Jorgensen, John Boak, Bill Spees, R.B. Sutton. **Lifetime Subscribers:** Michael S. Zick, F. Kirk.

Foreign Distributors: ASCII Publishing, Inc. (Japan), Computer Services (Australia), Computer Store (New Zealand), Computercollectief (Nederland), Homecomputer Vertriebs GMBH (West Germany), International Presse (West Germany), La Nacelle Bookstore (France), McGill's News Agency PTY LTD (Australia), Progressco (France).

People's Computer Company

Dr. Dobb's Journal is published by M&T Publishing, Inc. under license from People's Computer Company, 2682 Bishop Dr., Suite 107, San Ramon, CA 94583, a non-profit, educational corporation.

December 1984
Volume 9, Issue 12

CONTENTS

Masthead Changes

This special Unix issue was conceived by Reynold Wiggins but he was absent from the delivery room. Midterm, Reynold removed his editor's visor and threw away the nub of his red pencil to become a CAD programmer at Fairchild. So yours truly, Randy Sutherland, assisted the Doctor on this baby. Reynold Wiggins has been involved with *Dr. Dobb's Journal* almost since the beginning and we expect an ongoing relationship. When it comes to programming on microcomputers, it is hard to match his zeal!

New names will appear on the masthead next month: Frank DeRose, Assistant Editor; and Alex Ragen, Technical Editor.

Next Year

Next month we feature an article on how to "Fatten Your Mac" for a lot less than Apple's price. Also, we will review Logitech's Modula 2 compiler and STSC's APL interpreter. February is the Gala Anniversary Issue, 100 months of *DDJ*! In March we focus on artificial intelligence for microcomputers and announce the winner of the AI competition. April will feature human interface design and May will feature graphics algorithms. June will be the special telecommunications issue. Please submit manuscripts for the telecommunications issue by the end of February.

This Month's Referees

Dennis Allison, Stanford U.
S. M. Bellovin, AT&T
Wayne Chin, Hewlett-Packard
Bob Desinger, Hewlett-Packard
Mohammed El Lozy, Harvard U.
Jim Fleming, Unir Corporation

Angel Gomez, Telecomp, Inc.
John Keyes, Microsoft
Ben Laws, North Texas State U.
John Rogers, Fortune Systems
Joseph Sharp,
Micro Science Associates

STATEMENT OF OWNERSHIP, MANAGEMENT, AND CIRCULATION

(Act of August 12, 1970, Section 3685, Title 39, United States Code)

1. Title of Publication: Dr. Dobb's Journal, Publication No. 307690.
2. Date of Filing: October 12, 1984.
3. Frequency of Issue: Monthly (12 issues, \$25).
4. Location of Known Office of Publication: 2464 Embarcadero Way, Palo Alto, CA 94303.
5. Location of Headquarters of General Business Offices of the Publishers: 2464 Embarcadero Way, Palo Alto, CA 94303.
6. Names and Addresses of Publisher, Editor, and Managing Editor: Publisher, Laird Foshay, 2464 Embarcadero Way, Palo Alto, CA 94303. Editor, Michael Swaine, 2464 Embarcadero Way, Palo Alto, CA 94303. Managing Editor, Randy Sutherland, 2464 Embarcadero Way, Palo Alto, CA 94303.
7. Owner: Otmar Weber, 2464 Embarcadero Way, Palo Alto, CA 94303; and C. F. von Quadt, 2464 Embarcadero Way, Palo Alto, CA 94303.
8. Known Bondholders, Mortgagees, and Other Security Holders Owning or Holding 1 Percent or More of Total Amount of Bonds, Mortgages or Other Securities: None.
9. Extent and Nature of Circulation:

- A. Total number of copies printed: Average number of copies each issue during the preceding 12 months: 31,407. Actual number of copies of single issue published nearest to filing date: 35,550.
- B. Paid Circulation. 1. Sales through dealers and carriers, street vendors, and counter sales. Average number of copies each issue during preceding 12 months: 6,099. Actual number of copies of single issue published nearest to filing date: 6,625. 2. Mail subscrip-

tions. Average number of copies each issue during preceding 12 months: 18,268. Actual number of copies of single issue published nearest to filing date: 20,718.

- C. Total Paid Circulation. Average number of copies each issue during preceding 12 months: 24,367. Actual number of copies of single issue published nearest to filing date: 27,343.

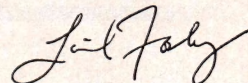
- D. Free distribution by mail, carrier, or other means, samples, complimentary, and other free copies. Average number of copies each issue during preceding 12 months: 230. Actual number of copies of single issue published nearest to filing date: 587.

- E. Total distribution. Average number of copies each issue during preceding 12 months: 24,597. Actual number of copies of single issue published nearest to filing date: 27,930.

- F. Copies not distributed. 1. Office use, left over, unaccounted, spoiled after printing. Average number of copies each issue during preceding 12 months: 1,820. Actual number of copies of single issue published nearest to filing date: 2,200. Returns from news agents. Average number of copies each issue during preceding 12 months: 4,990. Actual number of copies of single issue published nearest to filing date: 5,420.

- G. Total. Average number of copies each issue during preceding 12 months: 31,407. Actual number of copies of single issue published nearest to filing date: 35,550.

I certify that the statements made by me above are correct and complete.



Laird Foshay, Publisher

Dr. Dobb's Journal

ARTICLES

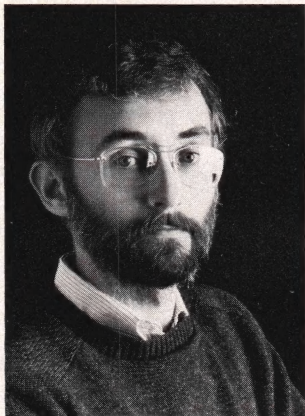
- | | | |
|---|-----------|---|
| Varieties of Unix
<i>by Alan Walworth</i> | 24 | A comparative overview of Unixes for microcomputers with a brief history of Unix and comments on its future, plus a guide to choosing a Unix (Reader Ballot No. 192) |
| Unix Device Drivers
<i>by John Bass</i> | 38 | Unix Version 7 drivers are the point of departure for this inside look at the Unix I/O subsystem and Unix device drivers (Reader Ballot No. 193) |
| A Unix Internals Bibliography
<i>by John Rogers</i> | 50 | An expert's guide to internals documentation so you won't have to "grep for it" (Reader Ballot No. 194) |
| A File Browser Program
<i>by John Johnson</i> | 60 | For those times when all you want to do is look through a file (Reader Ballot No. 195) |
| An Introduction to Parsing
<i>by Henry Seymour</i> | 78 | How to implement parsing schemes for assemblers, editors, or adventure games (Reader Ballot No. 196) |

COLUMNS

- | | | |
|---|------------|---|
| Dr. Dobb's Clinic
<i>by D. E. Cortesi</i> | 16 | Around and around with Shugarts and Tandons (Reader Ballot No. 190). |
| CP/M Exchange
<i>by Bob Blum</i> | 20 | The early days of CP/M; Using large sectors (Reader Ballot No. 191). |
| 16 Bit Software Toolbox
<i>by Ray Duncan</i> | 88 | Sneak 80286 Preview (Reader Ballot No. 197) |
| C/Unix Programmer's Notebook
<i>by Anthony Skjellum</i> | 96 | Long pointer corrections, programming philosophy (Reader Ballot No. 198) |
| Computer Calisthenics
<i>by Michael Wiesenber</i> | 120 | Puzzles (Reader Ballot No. 199) |

DEPARTMENTS

- | | | |
|--|------------|---|
| Editorial | 6 | |
| Letters | 8 | |
| Software Reviews | 106 | NCI Coherent, Turbo Pascal Version 2.0 (MSDOS and CP/M) |
| Of Interest
<i>by R. P. Sutherland</i> | 124 | (Reader Ballot No. 200) |
| Advertiser Index | 128 | |



Arguments for not doing a special issue on Unix: 1) I flew to New York on October 15 to attend a Unix Expo; I was not alone in finding the show slow and boring. (Well, not entirely boring; I did see some C programming tools that were of some interest, Jim Joyce's seminars seemed well-attended, and one company was showing off an intriguing product that implements Smalltalklike object-oriented programming in a Unix environment, but the show attendance was sub-moblevel.) 2) I drove up the peninsula to San Francisco on October 26-27 for the PC Faire; the Faire did nothing to reassure me that Unix is on the verge of taking over the exciting world of IBM PCs and compatibles. (OK, I did see some friendlifying frontends for PC Unixes.) 3) Dr. Dobb's readers have read all the inflated claims about Unix market share, don't believe them, and anyway don't read *DDJ* for that sort of thing; as one irascible newsletter editor who pontificates under the name Felgercarb N. Eloï recently wrote, "it was absurd to assert that Unix was or ever would be the best operating system for the dentists and candle-makers who comprise the mass personal computer market." (But many of our readers do use Unix.)

Argument for doing a special Unix issue: our readers are not dentists or candlemakers.

This special issue on Unix is a new departure for *DDJ*. We have in the past done special issues on Forth and telecommunications, and current plans call for us to continue doing these special issues. We will in fact be doing more special issues in the future. But we're changing our approach to them: instead of devoting an entire month of the magazine to a topic, we are now shining a more focused light on the topic at hand, leaving room for other articles in the same issue. The idea is that we're consciously structuring these special sections to deal in some depth with a relatively narrow slice of an interesting topic.

Consider the current issue's special section, which we have named Inside Unix. Here's what we think we have done here; you can let us know how well we succeeded. The Unix section includes only three articles (plus the regular "C/Unix" column and attention to Unix in "Of Interest"), and it centers on the Unix drivers article by John Bass.

The Bass article is directed at people who want something more than marketing hype on how many Unix licenses have been sold, or analysis of where Unix fits into the office of tomorrow. It takes you deeper into the Unix system, hence the section title Inside Unix.

Rogers' richly annotated bibliography of Unix internals is again not for the tyro but is a useful tool for the programmer with a need to know about Unix internals. We hope to supply such bibliographies whenever they are appropriate. The overview article by Walworth is here to provide a background for the other pieces and to clear up some confusion about versions of Unix for those who haven't got it all quite straight.

You're right if you're thinking that all this is some kind of experiment. As Castle, the slanderous stereotype of a humanities professor in B.F. Skinner's *Walden II*, said, the experimental attitude has the wonderful feature of letting one be utterly confident and self-righteous without knowing anything. We're experimenting, cautiously, with the magazine, and once we see the results of our present experiments, we'll probably continue experimenting.

Michael Swaine

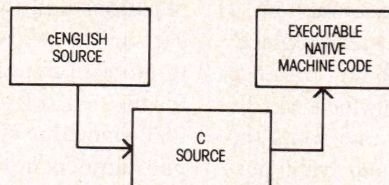
Michael Swaine

cENGLISH.™

The C Generation Language.

What is cENGLISH? cENGLISH is a comprehensive fourth generation procedural language based on dBASE II™ syntax. It is portable to a wide range of micros and minis. The language features user-transparent interfaces to a wide range of popular C compilers, operating systems, and data base managers.

How is portability achieved? cENGLISH through its compiler interface translates cENGLISH into documented C source and uses a host C compiler to produce native machine code.



Differences in the operating system and data base manager are handled by the runtime libraries.

The result is that cENGLISH source can be compiled without modification on any micro or mini configuration supporting cENGLISH.

What about performance? cENGLISH executes FAST, just like any compiled C program.

How easy is cENGLISH to use? While cENGLISH is a powerful high level language that can accommodate complex software development, it remains simple and straightforward to use.

Call or write for availability of cENGLISH for the following configurations—

Compilers:

Standard O/S compilers: Lattice C™ for MS/DOS™

Operating Systems:

UNIX™, UNIX-like, MS/DOS™, Coherent™, VMS™

Data Base Managers:

C-ISAM™ and INFORMIX™, UNIFY™, ORACLE™, PHACT™, Logix™

Foreign Language Versions:

German, French, Spanish

Attention MS/DOS users. Demo version and special introductory offer available for IBM PC, XT, AT, and other MS/DOS systems.

Requirements: 256K, hard disk or two floppy disk drives, and MS/DOS 2.1 or higher.

Attention dBASE II and dBASE III users. dBASE II to cENGLISH Converter now available; dBASE III Converter available later this quarter. Converted code is portable to micros or minis and executes as fast as original cENGLISH source.

dBASE II and dBASE III are trademarks of Ashton-Tate. Lattice is a trademark of Lattice. Inc. UNIX is a trademark of Bell Laboratories. MS/DOS is a trademark of Microsoft, Inc. Coherent is a trademark of Mark Williams Company. VMS is a trademark of Digital Equipment Corporation. C-ISAM and INFORMIX are trademarks of Relational Database Systems, Inc. Oracle is a trademark of Oracle Inc. PHACT is a trademark of Phact Associates. Logix is a trademark of Logical Software, Inc. IBM PC XT and AT are trademarks of International Business Machines Corporation. UNIFY is a trademark of Unity Corp.

SAMPLE cENGLISH PROGRAM

IDENTIFICATIONS

MODULE: Mininame

AUTHOR: bcs

DATE: 8/29/84

REMARKS: Sample cENGLISH program that adds first names to a file

END IDENTIFICATIONS

GLOBS

FIXED LENGTH 1 ans

FIXED LENGTH 15 Fname

END GLOBS

MAIN PROGRAM

BEGIN

CLEAR SCREEN

SET ECHO OFF

USE "NAMES"

VIEW BY "ID_FNAME" ASCENDING

AT 23.1 SAY "Add a record? Y or N"

AT 23.25 ENTER ans USING "I"

WHILE ans EQ "Y"

CLEAR GETS

AT 6.1 SAY "Enter first name"

AT 6.20 GET Fname

READ SCREEN

INSERT

Fname = Fname

END INSERT

AT 12.10 SAY "Welcome to cENGLISH" & Fname

WAIT

AT 14.10 SAY "HIT ANY KEY TO CONTINUE"

STORE " " TO Fname

STORE " " TO ans

AT 23.1 SAY "Add another record? Y or N"

AT 23.30 ENTER ans USING "I"

CLEAR ROW 1 THRU 23

END WHILE

AT 12.10 SAY "That's all for now!"

UNUSE "NAMES"

SET ECHO ON

END PROGRAM

**I'd like to know more about cENGLISH.
Please send further information.**

Your Name _____ Title _____

Company _____ Telephone _____

Address _____

City _____ State _____ Zip _____

Check one: ☐ End User ☐ System House ☐ Dealer ☐ Distributor

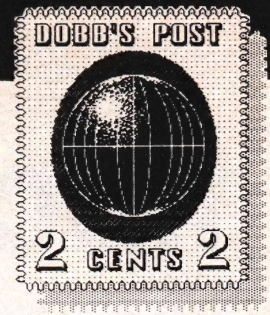
Send to: cLINE Inc., 20 West Ontario, Chicago, IL 60610-3809

Telex 516315 Phone (312) 944-4510

In Canada: cLINE Canada, Inc. Complexe La Laurentienne,
425 St. Amable, Suite 105, Quebec, Canada G1R5E4
Phone (418) 524-4641

F4Q84

cLINE™, Inc.



Forum

Dear DDJ,

I was interested to see Alex Cameron's contribution of standard (K&R) fopen/fclose functions for BDSC in the August '84 *Dr. Dobb's Journal* ("BDSC Runtime Solution," page 118). I contributed code with a similar goal to the C User's Group over two years ago. Cameron took the conventional approach of using alloc/free to allocate buffer space in the "heap," the region of memory between the top of the externals and the bottom of the stack. I was more concerned to avoid the tendency to allocate too many buffers and bash the stack, so I #defined the maximum number of buffers (simultaneously open files) as NIOBUFS before main(), and reserved the buffers in the external data space. Thus, inadequate memory problems would be obvious at link time. My code was set up to be #included at three points in main(). I have recently ported eleven substantial programs from BDSC to Computer Innovations C86 and found the use of standard fopen/fclose to be a real help.

More recently, along with a text formatter which I wrote, I contributed a library of 43 general purpose functions to the C User's Group ("Martz Library Disk"). Your readers writing in C may save some time by getting a copy of these. Most of the functions are for character string manipulation. For example, argmatch() finds arguments to main() in any sequence, with leading/trailing ambiguities, optionally deleting them from the argc, argv list once found. fbrkout() breaks a long string into pieces of specified width and hands it to an output stream. Breaks are between words. It has options for indentation and for representing non-printable characters. badname() verifies that a filename is valid according to CP/M file-naming rules; if not, it issues a de-

tailed error message. findwords() counts the number of words in a string and sets up an array of pointers pointing to each. pack() packs strings into a big buffer for later retrieval, returning a pointer to the beginning of each. substitute() replaces all instances of a specified string with a specified replacement. todelim() finds the first instance of a specified delimiter (which can be one or more characters) and splits the original string into left and right portions excluding the delimiter. ynqd() is one of the most useful: "Yes No Question with Default." As an example of its use, to ask whether the user needs help:

```
if (ynqd("Do you need help", YES))
    givehelp( );
```

This displays on the console:

```
Do you need help? (y/n)
(default = YES)
```

Simply hitting a carriage return [signals] the default YES In addition to my own modest contributions, many others have contributed tools and programs totaling about forty 8-inch SSSD CP/M disks. Readers programming in C should avail themselves of these by writing to the C User's Group, 112 N. Main, Box 287, Yates Center, KS 66783; phone 316-625-3553. Six issues of the CUG Newsletter are \$10, and 8-inch SSSD CP/M disks are only \$8. Several 5-inch disk formats are also available (Apple II, Heath/Zenith, TRS-80, Northstar, Osborne, and others).

Sincerely,
Eric Martz, Ph.D.
48 Hunter's Hill Circle
Amherst, MA 01002

Dear DDJ,

I read the article by Joe Barnhart about the fast Fourier transform (FFT) in the September 1984 issue. A disad-

vantage of the FFT is that all the numbers in the transform are complex, requiring complex arithmetic. A complex addition requires two additions of real numbers, and a complex multiplication requires four real multiplications and two real additions. The user has to use twice as much memory to store a complex array of numbers for the FFT. I know of variations of the FFT algorithm that repack the real input numbers into a complex array half the original length. The FFT is performed and the result unpacked. The FFT of the repacked array uses less operations. However, the packing and unpacking algorithms are complicated.

An alternative to the FFT is the fast Hartley transform (FHT). The Hartley transform is similar to the Fourier transform, except that the input and output numbers remain real. Since most applications of the Fourier transform usually involve real input data, the FHT is more suitable for the average user. Since the FHT does not involve any complex operations, the FHT requires fewer multiplications and additions than the FFT algorithm presented in the article. The basic equation for N-point Fourier transform is

$$X_f(k) = \sum_{n=0}^{N-1} x(n) (\cos((2\pi/N)nk) - i \sin((2\pi/N)nk))$$

where $x(n)$ is the complex input, $X_f(k)$ is the Fourier transform, and i is the square root of -1 . The equation for the N -point Hartley transform is

$$X_h(k) = \sum_{n=0}^{N-1} x(n) (\cos((2\pi/N)nk) + \sin((2\pi/N)nk))$$

where $x(n)$ is the real input and $X_h(k)$ is the Hartley transform. Note that the

Hartley transform does not have any complex arithmetic involved. The inverse Fourier transform is different from the forward Fourier transform. The inverse Hartley transform is the same as the forward Hartley transform. The FFT makes use of the symmetry of the Fourier transform, which leads to the butterfly graph shown in the article. The Hartley transform also has a symmetry which leads to a FHT algorithm that uses a double butterfly graph. The user can switch the results from a Hartley transform to the Fourier transform through the equations

$$\begin{aligned} \text{REAL}[X_f(k)] \\ = [X_h(k) + X_h(-k)]/2 \end{aligned}$$

$$\begin{aligned} \text{IMAG}[X_f(k)] \\ = [X_h(k) - X_h(-k)]/2 \end{aligned}$$

Enclosed [see Listing One on page 12] is a C listing of a subroutine that performs a FHT. Many common operations of the frequency domain are computed faster in the Hartley domain. For example, convolution and cross correlation require fewer operations when done with the Hartley transform.

I hope you will let your readers know about the Hartley transform, because it is better suited for small computer systems than the Fourier transform. For a good reference, see Ronald Bracewell's article, "The Fast Hartley Transform" in the August 1984 issue of *Proceedings of the IEEE*. Another reference is my article "An Algorithm for the Fast Hartley Transform" in the Stanford Exploration Project #38.

Sincerely,
Ron Ullmann
Picture Element Ltd.
635 Waverly
Palo Alto, CA 94301

Dear Doctor:

In a recent issue, a reader told you about how a patch to the DRI macro-assembler RMAC, which he received from Manx, solved his need to use underscores and periods in identifiers. Anyone mixing C or Pascal with assembly language modules linked together will find this enhancement to RMAC useful.

Because the above-mentioned patch may have wide appeal to your audience of system hackers, I am including a list-

Still Fixing Bugs The Hard Way?



Ready to take the sting out of debugging? You can with Pfix86™ and Pfix86 Plus™, the most advanced dynamic and symbolic debuggers on the market today for PC DOS and MS-DOS™ programmers.

What other debugger offers you an adjustable multiple-window display so you can view program code and data, breakpoint settings, current machine register and stack contents all at the same time? And, an inline assembler so you can make program corrections directly in assembly language. Plus, powerful breakpoint features that allow you to run a program at full speed until a loop has been performed 100 times, or have the program automatically jump to a temporary patch area.

Or maybe you're tired of searching through endless piles of listings for errors? With Pfix86 Plus you won't have to. You can

locate instruction and data by the symbolic name and using the symbolic address. Handle larger, overlaid programs with ease. And, Pfix86 Plus is designed to work with our Plink86™ linkage editor.

But that's not all. With a single keystroke you can trace an instruction and the action will be immediately reflected in code, data, stack, and register windows. Pressing a different key will elicit a special trace mode that executes call and loop instructions at full speed, as though only a single instruction were being executed.

And you get an easily accessible menu that makes the power of our debuggers instantly available to the new user, but won't inhibit the practiced user.

So, why struggle with bugs? Pfix86 by Phoenix. Pfix86 \$195. Pfix86 Plus \$395.

Call (800) 344-7200, or write.

Phoenix

Phoenix Computer Products Corporation

1416 Providence Highway, Suite 220
Norwood, MA 02062
In Massachusetts (617) 762-5030

Pfix86, Pfix86 Plus and Plink86 are trademarks of Phoenix Software Associates Ltd.
MS-DOS is a trademark of Microsoft Corporation

AT LAST

S-100 ↔ 488

THAT

DOES

EVERYTHING

YOU WANT

IT TO DO



D&W DIGITAL, INC.
20655 Hathaway Avenue
Hayward, California 94541
(415) 887-5711

ing [see Listing Two on page 13] that achieves the purpose, and which can also be customized for MAC. This same listing was available from several bulletin boards, but given the huge amount of software that those boards carry, it might have been unnoticed by many.

I would like to point out that any potential commercial firm interested in this patch should make arrangements with me prior to using it.

In an entirely different matter, Ray Duncan and thereafter some readers discovered many problems with MASM, the Microsoft/IBM 8086 macroassembler. It's a sorry state of affairs that some companies can provide such low level of quality to the marketplace and still be praised by much of the press. But the reason I'm writing you about this is that I'm using two versions of MASM; one is 1.07 and the other is 2.04. The former one seems to have less problems than the 2.04. Many of the bugs found by Mr. Duncan are not present in version 1.07, which seems to perform normally in these tests. I would recommend readers to try to obtain this version and see if I'm right.

Some other bugs are still present, even in this better version, plus problems resulting from being a two-pass assembler. I don't know or have RASM, but ASM-86, which comes with CP/M86, is a three-pass assembler, which allows it to do a better allocation of space before actually starting the assembly. This eliminates many nonsense NOPs that MASM needs to scatter through the object code, degrading even more a slow-performing chip like the 8088.

Regards,
George Blat
Blat Research +
Development Corp.
8016 188th Street SW
Edmonds, WA 98020

Dear DDJ,
"Designing a File Encryption System" (DDJ, August 1984) by Thomas and Thersites is delightful. Can you imagine the hilarity in the halls at the Puzzle Palace when their cafeteria serves recycled permutation table generators? I can just see the FBI on stakeout at the embassy supermarkets watching for a surge in sales of one pound bags

of leguminous encryption aids.

Sincerely,
Adam Fritz
133 Main Street
Afton, New York 13730

DRI Support

Dear Editor:

I am writing in response to Steve Conley's letter in the August issue of *Dr. Dobb's Journal*. Mr. Conley described a bug he found in RMAC. He was upset when people at Digital Research told him that they didn't know when, or if, the problem would be fixed. I would like to address several of the concerns which he raised.

It has always been Digital Research's policy to be honest with our customers. I truly wish we could fix all the bugs in all of our products as fast as we would like. Unfortunately, the reality is that this doesn't happen. Instead we try to prioritize problems and do all we can to make sure our customers are aware of them. Fixing this problem has a low priority because it is possible to work around the problem and because RMAC is an 8-bit product. If we had promised Mr. Conley that the problem would be fixed "soon" he probably would have been satisfied. I firmly believe, however, that it is much better to be honest. Therefore the engineer described the status of the problem to Mr. Conley and discussed with him two possible work arounds for the problem. To me this represents quality support rather than indifference.

Providing quality technical support to a large customer base can be difficult. The method which we have chosen to provide this support is our Professional Programmer Support Program (PPS). PPS includes unlimited toll-free phone access to our engineers, a technical newsletter and a subscription to CompuServe so that the customer can access our data bases. This package is available for only \$250 per year for each customer contact person.

I think that PPS is what Mr. Conley was referring to when he mentioned a \$250 software maintenance package. Maintenance, however, is available to all registered users of our products. We use the registered users data base to notify customers whenever a new re-

Available
for IBM PC

What C did for Programming

Mark Williams has done for C Programming

The C Programming System from Mark Williams

MWC86 gets your C programs running faster and uses less memory space than any other compiler on the market. Then *csd*, Mark Williams' revolutionary C Source Debugger, helps you debug faster. That's The C Programming System from Mark Williams Company.

MWC86

MWC86 is the most highly optimized C compiler available anywhere for the DOS and 8086 environment. The benchmarks prove it! They show MWC86 is unmatched in speed and code density.

MWC86 supports large and small models of compilation, the 8087 math coprocessor and DOS 2.0 pathnames. The compiler features common code elimination, peephole optimization and register variables. It includes the most complete libraries. Unlike its competition, MWC86 supports the full C language including recent extensions such as the Berkeley structure rules, voids, enumerated data types, UNIX* I/O calls and structure assignments.

Quality is why Intel, DEC and Wang chose to distribute MWC86. These industry leaders looked and compared and found Mark Williams to be best.

User Friendly

MWC86 is the easiest to use of all compilers. One command runs all phases from pre-processor to assembler and linker. MWC86 eliminates the need to search for error messages in the back of a manual. All error messages appear on the screen in English.

A recent review of MWC86 in *PC World*, June, 1984, summed it up:

"Of all the compilers reviewed, MWC86 would be my first choice for product development. It compiles quickly, produces superior error messages, and generates quick, compact object code. The library is small and fast and closely follows the industry standard for C libraries."

csd C Source Debugger

Mark Williams was not content to write the best C compiler on the market. To advance the state of the art in software development, Mark Williams wrote *csd*.

csd C Source Debugger serves as a microscope on the program. Any C expression can be entered and evaluated. With *csd* a programmer can set breakpoints on variables and expressions with full history capability and can single step a program to find bugs. The debugger does not affect either code size or execution time. *csd* features online help instructions; the ability to walk through the stack; the debugging of graphics programs without disturb-

ing the program under test; and evaluation, source, program and history windows.

csd eases the most difficult part of development — debugging. Because *csd* debugs in C, not assembler, a programmer no longer has to rely on old-fashioned assembler tools, but can work as if using a C interpreter — in real time.

The C Programming System from Mark Williams now supports the following libraries:

Library	Company
Windows for C	Creative Solutions
Halo	Media Cybernetics
PHACT	PHACT Associates
The Greenleaf Functions	Greenleaf Software
Btrieve	SoftCraft

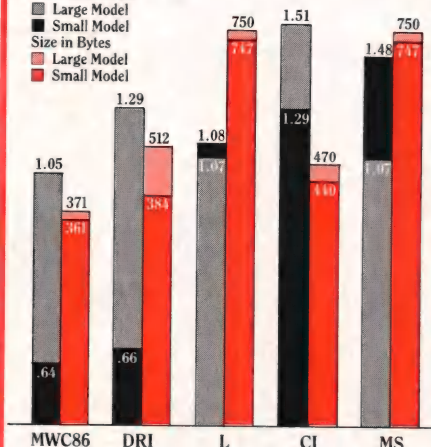
The C Programming System from Mark Williams

The C Programming System from Mark Williams delivers not only the best C compiler for the 8086 but also the only C source level debugger. That's why it does for C programming what C did for programming. The Mark Williams C Programming System gives the programmer the MWC86 C compiler and the *csd* C Source Debugger for only \$495. Order today by calling 1-800-MWC-1700. Major credit cards accepted.

Technical support for The Mark Williams C Programming System is provided free of charge by the team that developed it.

SIEVE

Time in Seconds
■ Large Model
■ Small Model
Size in Bytes
■ Large Model
■ Small Model



*Unix is a Trademark of Bell Laboratories.



Mark Williams Company
1430 W. Wrightwood Ave.
Chicago, IL 60614

lease is available. Unfortunately we could not find Mr. Conley at the address he listed as either a subscriber to PPS or as a registered user. Because of this, we will not be able to notify him when updates occur for any of his DRI products.

Mr. Conley mentioned problems he has had reaching DRI by phone. Recently we have made some improvements so that it is now relatively easy to reach our Warranty Support Department. They do not provide customers with technical support but do provide customers with warranty support. Their number is (408) 646-6464.

In summary, I am very proud of the support we, at DRI, provide our customers. I believe we have found a very good way to provide a much needed service to our customers.

Sincerely,
Marion Brown
Support Center Manager

160 Central Avenue
Pacific Grove, CA 93950

Caveat Emptor

Dear DDJ,
The purpose of this letter is to warn *Dr. Dobb's* readers of problems in dealing with JRT.

Early in June of this year I called JRT on the phone and ordered a copy of their Modula-2. I was told that the product would be shipped within a week. On 20 June my VISA account was billed \$102.95 for Modula-2 plus shipping.

When the product had not arrived by August, I tried over a period of three weeks, repeatedly and at all hours, to reach JRT at their listed phone. The phone was never answered. Finally, at the end of August I sent JRT a certified letter, return receipt requested, describing the situation and

asking that either my copy of Modula-2 be shipped or that my VISA account be credited. The return receipt came back to me on 7 September, signed by Jim Tyson of JRT.

It is now 24 September and I have heard nothing from JRT, I have not received Modula, and my VISA account has not been credited. While most of my experience ordering from computer suppliers has been extremely satisfactory, I think JRT's actions, taking your money and not shipping the product, are at best a sharp business practice and, at worst, border on fraud.

Sincerely,
R. A. Langevin
7621 Fontaine Street
Potomac, MD 20854

DDJ

Letters (Text begins on page 8)

Listing One

Fast Hartley Transform

```
/* The FHT performs the fast Hartley transform over an array of
 * floating point numbers. The array of real numbers are pointed to by
 * FX. LENGTH is the number of points in the array and must be a power of
 * two (i.e. 16, 32, 2048). The Hartley transform of the array is stored
 * in the array FX upon completion. If LENGTH is not a power of two, FHT
 * returns a value of -1. Otherwise, FHT returns 0. By Ron Ullmann
 */
(length, fx)
int length; double *fx;
{
    int ii, kk, jj, ll, istep;
    double *pba, *pbb, *pbc, *pbd;
    double temp1, temp2, arg, fcos, fsin, dsin, dcos;

    /* Test to see if length is a power of two and is not zero. */
    for (kk = length; (kk & 1) == 0; kk >>= 1);
    if (kk != 1) return (-1);
    /* Reorder the data */
    arg = sqrt (1. / (double) length); /* scale the data */
    jj = 0;
    for (ii = 0, kk = 0; ii < length; ++ii, kk += jj)
    {
        if (ii <= kk)
        {
            temp1 = *(fx + kk) * arg;
            *(fx + kk) = *(fx + ii) * arg;
            *(fx + ii) = temp1;
        }
        for (jj = length>>1; kk >= jj && jj >= 1; kk -= jj, jj >>= 1);
    }
    arg = 3.141592653589793238462643;
    for (jj = 1; jj < length; jj = istep)
    {
        istep = jj << 1;
        dcos = cos (arg); dsin = sin (arg);
        arg /= 2.;
        for (ii = 0; ii < length; ii += istep)
        {
            pba = fx + ii; pbc = pba + jj;
            temp1 = *pbc;
            *pbc = *pba - temp1; *pba += temp1;
        }
    }
}
```



```

    }
    fcos = dcos; fsin = dsin;
    for (ll = 1, kk = jj - 2; ll < (jj >> 1); ++ll, kk -= 2)
    {
        for (ii = ll; ii < length; ii += istep)
        {
            pba = fx + ii; pbb = pba + kk;
            pbc = pba + jj; pbd = pbc + kk;
            temp1 = fcos * *pbc + fsin * *pbd;
            temp2 = fsin * *pbc - fcos * *pbd;
            *pbc = *pba - temp1; *pbd = *pbb - temp2;
            *pba += temp1; *pbb += temp2;
        }
        temp1 = fcos*dcos - fsin*dsin;
        fsin = fsin*dcos + fcos*dsin; fcos = temp1;
    }
    if (jj > 1)
        for (ii = (jj >> 1); ii < length; ii += istep)
        {
            pba = fx + ii; pbc = pba + jj;
            temp1 = *pbc; *pbc = *pba - temp1;
            *pba += temp1;
        }
    }
    return (0);
}
/* end of FHT */

```

End Listing One

Listing Two

```

;*****
;
;           Patches for MAC and RMAC
;-----
;
;           by George Blat
;   Blat, Research + Development Corp.
;           8016 188th SW
;           Edmonds, WA 98020
;
;*****
;
;The following changes are (c)1983 Blat R+D Corp. Permission is
;granted to use these patches only in non-commercial applications.
;MAC and RMAC are trademarks of Digital Research, Inc. which holds
;ownership and all rights to the original programs.
;
;*****
;
;Mac and Rmac are two reliable assemblers developed by Digital
;Research which have a good number of useful features. It seems
;natural to get the most out of them.
;
;Among the features that can be added to Mac and Rmac, are the
;ability to use the period '.' and the underscore '_' as part of
;symbol names such as labels, even as first character of the
;symbol. The underscore, for instance, makes a much better word
;separator than the dollar '$' sign when used in a multi-word
;label. In a dense program listing, it's certainly easier to find
;STAT_PORT than STAT$PORT, and @hl_to_de than @hl$to$de.
;
;By the same token, I don't agree with the decision of Digital
;Research of making the dollar sign a don't care character. It
;introduces confusion as it allows symbols that don't look the
;same to be equivalent.
;
;In addition, RMAC can be easily patched to create .REL files
;where the global (external) names have up to 7 active characters.
;This helps by allowing you to create more meaningful symbol names
;and therefore improve program legibility. This change is still
;entirely compatible with the industry standard Microsoft format.
;
;The following patches should be assembled with MAC (not RMAC)
;and the resulting hex file should be applied over the original
;programs with DDT, SID or ZSID. KEEP AN ARCHIVE COPY OF THE
;ORIGINAL MAC OR RMAC BEFORE PATCHING.

```

```

false equ 0
true equ not false

```

(Continued on next page)

Letters (Listing Continued, text begins on page 8)

Listing Two

```
rmac      equ      true          ;select one and only one of these
mac       equ      false        ;true

      if      rmac
global7   equ      true          ;set to false if you don't want
                                   ;7 char globals
patcharea      equ      13bh
dollarcounts   equ      1d7ah    ;set this to false if you like to
                                   ;keep the dollar as a don't care char
checkalfa      equ      1d9ch
toup           equ      2844h
      endif

      if      mac
copyright     equ      103h      ;shorten but keep the copyright notice
dollarcounts  equ      1834h
checkalfa     equ      1853h
      endif

      if      mac
      org     copyright
      db      '(c)1977 DRI'
patcharea:
      endif

      if      rmac
      org     patcharea
      endif

      cpi     '.'
      rz
      cpi     '_'
      rz
      cpi     '?'
      rz
      cpi     '@'
      rz
      if      rmac
      call    toup
      endif
      sui     'A'
      cpi     'Z'-'A'+1
      cmc
      ret

      if      rmac and global7
compare    equ      12d6h
setit7     equ      12dbh

      org     compare
      cpi     8          ;replaces cpi 7
      org     setit7
      mvi     a,7        ;replaces mvi a,6
      endif

      if      dollarcounts
      org     dollarcounts
      nop                    ;replaces mov m,a
      endif

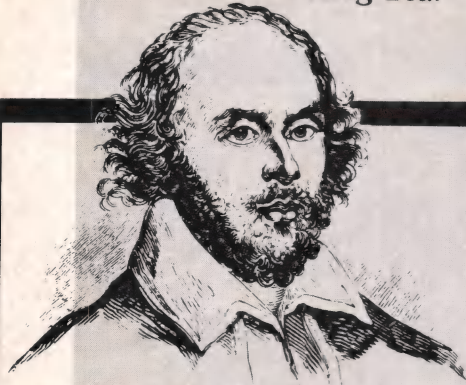
      org     checkalfa
      call    patcharea    ;replaces cpi 3f
      cmc                    ;jz      ldbl

      sbb     a            ;cpi     40
      ret                    ;jz      ldbl,   etc.

      end
```

End Listing Two

At Christmas I no more
desire a rose
Than wish a snow in May's
new fangled mirth
But like of each thing
that in season grows
—King Lear



MacInker

A Gift For Christmas A Gift For All Seasons

If Shakespeare had had a word processor he would have consumed about 25 cartridges to run a first draft of his works. At an average of \$10/cartridge the cost is \$250. With MAC INKER he would use one cartridge, his total would be 50 cents in ink and his print-out quality would be much improved.

MAC INKER is very simple to use and automatic. Average ink cost/re-inking is 5 cents. We support 535 printers and we have 20,000 units in the field, in the US and in 5 continents.

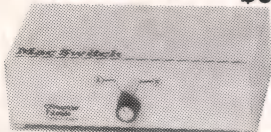
MAC INKER, a gift for Christmas, that will last for years in many seasons to come.

\$54.95+



MacSwitch

Choose also our popular MAC SWITCH, serial or parallel switch - the ideal companion for the user who has 2 printers or 2 microcomputers or both. **\$99.00**



Order toll free 1-800-547-3303
or ask for free brochure

Computer Friends

6415 S.W. Canyon Court
Suite #10
Portland, Oregon 97221
(503) 297-2321

The Preferred C Compiler

"...C86 was the only compiler we tested that ran every benchmark we tried and gave the expected results... Computer Innovations C86 was the compiler that our staff programmers used both before and six months after we conducted the tests."

J. Houston, BYTE MAGAZINE - February 1984

*FAST EXECUTION -
of your programs.

*FULL & STANDARD
IMPLEMENTATION OF C -
includes all the features described by
K & R. It works with the standard
MSDOS Linker and Assembler; many
programs written under UNIX can
often be compiled with no changes.

*8087 IN-LINE -
highly optimized code provides 8087
performance about as fast as possible.

*POWERFUL OPTIONS -
include DOS2 and DOS1 support and
interfaces; graphics interface capability;
object code; and librarian.

*FULL LIBRARY WITH SOURCE -
6 source libraries with full source code
the "large" and "small" models, soft-
ware and 8087 floating point, DOS2
and DOSALL.

*FULL RANGE OF SUPPORT
PRODUCTS FROM COMPUTER
INNOVATIONS -
including Halo Graphics, Phact File
Management, Panel Screen Manage-
ment, C Helper Utilities and our
newest C to dBase development
tool.

*HIGH RELIABILITY -
time proven through thousands of
users.

*DIRECT TECHNICAL
SUPPORT -
from 9 a.m. to 6 p.m.

Join The Professional Programmers Who Agree C86™ Is The C Compiler Of Choice

For Further Information Or To Order Call:

800-922-0169

Technical Support: (201) 542-5920

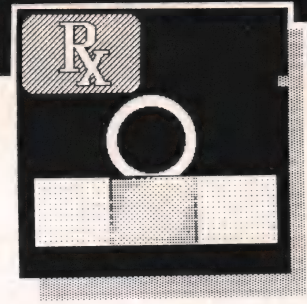
980 Shrewsbury Avenue
Suite PW509
Tinton Falls, NJ 07724



Computer Innovations, Inc.

C86™

PRICES SUBJECT TO CHANGE WITHOUT PRIOR NOTICE
UNIX IS A TRADEMARK OF BELL LABS C86 IS A TRADEMARK OF COMPUTER INNOVATIONS, INC. MSDOS IS A TRADEMARK OF MICROSOFT
PCDOS IS A TRADEMARK OF INTERNATIONAL BUSINESS MACHINES



by D. E. Cortesi, Resident Intern

The Intern Loses His Head: A Cautionary Tale

It was a dark and stormy night. Rain lashed against the windows of the Clinic. The Intern sprawled across a gurney. He pushed his mask aside and slurped from a paper cup of coffee. Winced. Sighed. And spoke, his voice barely audible above the wind. "I lost a patient," he said. "It was my own damn fault, sheer arrogance. But I paid for it . . ."

The New Drives

I had a pair of Shugart 800s (he went on), single-sided 8-inch drives. They worked fine, but I wanted more. Six hundred Kb just isn't enough for some of the things I do. Double-sided drives have 1200K, and with 256 directory entries they'd hold any project short of the U.S. Census.

And they're much faster—you know me and speed. Modern drives step at 3 milliseconds, twice as fast as my old drives, and with twice the data under the heads they incur fewer and shorter seeks on the average.

But I wanted not merely new drives but half-height drives. No practical reason—just because they're sexy. Oh, a pair of half-heights is a lot less bulky than two normal drives, but my real motive was . . . aesthetic.

So I placed an order with Floppy Disk Services (741 Alexander Road, Princeton, NJ 08540; 800-223-0306) for an enclosure with power supply, two drives, and a cable. It cost just over \$1300.

Floppy Disk Services will configure just about any combination of drives, and I had to choose a make of drive. Here was the first place I really went wrong. I asked for Shugart 860s for no better reason than that the old Shugarts had given such good service. If I'd only asked for advice—from FDS

or a local guru—I wouldn't have ended up holding a detached head in my hand.

The Interface

The disks came a month later. The enclosure was handsome, the power supply looked sturdy, and the drives were . . . aesthetically delicious. Open cut-outs on the back panel might have allowed air to shortcut the fan, but a piece of cardboard from the shipping carton blocked them neatly.

I knew that all drives have a zillion jumper options and that the odds against FDS plugging them right for my system were astronomical, so I'd ordered the Shugart manual. That was a good move. The manual made it glaringly obvious that floppy technology had changed quite a bit since my disk controller, a venerable CCS 2422, had been designed.

My first problem was how to control the drive motors. The old drives rotated all the time but only closed the heads onto the disk when the 1793 asserted the head-load line (clack!). When five index holes passed without any activity, the 1793 would automatically reset the head-load (clink!) so the disk could turn freely.

The new drives loaded the heads as soon as a disk was inserted. The way their options were strapped on arrival, they would rotate the disk as long as the drive-select line was asserted. My BIOS (which followed the original CCS BIOS) never cleared the drive-select port, so the last-used drive would rotate indefinitely—with the heads pressing on the disk. Not good.

The new drives allowed a jumper option that would start the motor only when the drive was selected *and* the head-load signal was asserted. Another option would keep it turning for 5 seconds after the fall of head-load then stop it. These options solved the first

problem very nicely.

They also created the second problem. The new drives might take 168 milliseconds to come up to speed after head-load was asserted. On the other hand, they *might* be ready to go instantly—if the motor were still turning from the last access. How could the 1793 controller chip know when to wait for the motor to start and when it needn't wait?

Well, these drives emit a signal that was new to me. The line, True Ready, is pin 8 of the interface. The drive asserts it when the motor is up to speed and the head is stable; this is exactly what the 1793 chip needs to tell it to go ahead with a read or write. What's more, the 1793 has an input, Head-Load Timing, that takes exactly that information. Unfortunately, pin 8 on my hoary old disk controller board is a no-connect. The Head-Load Timing input to the 1793 is developed by a 50 millisecond one-shot on the board itself.

OK, I told myself, that's why I own an X-acto knife and a soldering iron, isn't it? It took most of a weekend to work out that it is possible, using only existing components of the CCS 2422 board, to implement True Ready as the Head-Load Timing signal of the 1793. And it worked. I got the Shugart 860s reading and writing with all four heads.

The Decapitation

After a few days, drive B started to fail intermittently. Sometimes it was "record not found"; sometimes it was a CRC error on a sector ID; sometimes an error on a sector proper. Strange thing, it was worse with Dysan disks than with junk disks.

OK, now I had a choice. I could refit the old drives, pack up the new ones, cart them to the UPS office, pay to ship them back to New Jersey, and wait (how many weeks?) for them to come back. Or I could take off the cover and

have a peek myself. One little peek wouldn't hurt.

With the covers off, I could *hear* the problem. The rotational speed was varying enough to be audible. Was it a bad motor? No! There was simply too much pressure on the disk jacket. The drive has a spring-loaded shoe that bears down on the disk jacket. I put out a cautious pinky and lifted the shoe: the disk sped up and a bad sector became readable. I released it: the drive slowed down audibly and an error occurred. Maybe the Dysan disks had thicker jackets or a little more internal friction.

This is the point at which I lost the patient. I looked into the drive and decided that the pressure of the shoe was set by the position of a big plastic cam. The same cam raised the head and tripped the eject arm when the drive was opened. Its position was maintained by two setscrews. They were covered with touch-me-not varnish. Should I ship the drives back, or should I get out a screwdriver and try an adjustment? You can guess what I did.

The cam adjustment was tricky. The cam worked against a heavy torsional spring that fought every move. Worse, after I started tinkering, it dawned on me that it wasn't the cam at all. The pressure shoe just floated in it, held down by a weentsy little spring of its own. I got the cam back where it should be, more or less, fixed the shoe pressure, and buttoned it up. And it worked! My goodness, was I relieved. Maybe I had flubbed a little, I told myself, but, by golly, I had diagnosed and repaired the problem.

Ah, the bliss of ignorance. That big spring was working away on the head-lift cam, gradually shifting the setscrews. The head was lifting less and less when the disk was ejected. I noticed a little extra noise when the disk popped out of the drive but thought nothing of it. Until the drive quit working.

Catastrophe!

It wouldn't read or write on side one. I took the cover off again and looked inside, and a bowling ball dropped into my stomach. The upper head was dangling on its fragile wires, completely detached from the arm!

A quick look with a dental mirror revealed the awful truth. In the Shugart

860, the upper head is suspended in a plastic frame. The sum total of its support is a pair of tiny bronze straps. They are less than a millimeter wide and no thicker than a sheet of paper. If the head doesn't lift far enough, the edge of the disk jacket will tweak it as the disk pops out. The head will flop back and forth on these flimsy metal straps. Do it often enough, the straps will fail and the head will dangle on its wires.

OK, there I was with a thoroughly inop drive and not a prayer of making a warranty claim. So I started making phone calls, trying to find a professional to fix the drive. What I got was more bad news bulletins than a stock ticker

in 1927.

The first guy I talked to told me that, although Shugart 800s were "built like Mac trucks, they run forever," the 860 half-heights were "uh, let's say, a little fragile. You shoulda got the Tandon 848-2, it's a real nice drive." Thanks for the tip; what about replacing the upper head?

No, the only repair unit for the heads was the entire head assembly, upper and lower both, and it cost just under \$300. Fine, I said, that's less than a whole drive, when can we get one?

That could be a problem, I was told. "We think of Shugart as being three miles and thirty days away. We might

CP/M-80 C Programmers . . .

Save time

. . . with the BDS C Compiler. Compile, link and execute *faster* than you ever thought possible!

If you're a C language programmer whose patience is wearing thin, who wants to spend your valuable time *programming* instead of twiddling your thumbs waiting for slow compilers, who just wants to work *fast*, then it's

time you programmed with the BDS C Compiler.

BDS C is designed for CP/M-80 and provides users with quick, clean software development with emphasis on systems programming.

BDS C features include:

- Ultra-fast compilation, linkage and execution that produce directly executable 8080/8085 CP/M command files.
- A comprehensive debugger that traces program execution and interactively displays both local and external variables by name and proper type.
- Dynamic overlays that allow for run-time segmentation of programs too large to fit into memory.
- A 120-function library written in both C and assembly language with full source code.
- Plus . . .
- A thorough, easy-to-read, 181-page user's manual complete with tutorials, hints, error messages and an easy-to-use index — it's the perfect manual for the beginner and the seasoned professional.
- An attractive selection of sample programs, including MODEM-compatible telecommunications, CP/M system utilities, games and more.
- A nationwide BDS C User's Group (\$10 membership fee — application included with package) that offers a newsletter, BDS C updates and access to public domain C utilities.

Reviewers everywhere have praised BDS C for its elegant operation and optimal use of CP/M resources. Above all, BDS C has been hailed for its remarkable speed.

BYTE Magazine placed BDS C ahead of all other 8080/8085 C compilers tested for fastest object-code execution with all available speed-up options in use. In addition, BDS C's speed of compilation was almost *twice* as

fast as its closet competitor (benchmark for this test was the Sieve of Eratosthenes).

"I recommend both the language and the implementation by BDS very highly."

Tim Pugh, Jr.
in *Infoworld*

"Performance: Excellent.
Documentation: Excellent.
Ease of Use: Excellent."

InfoWorld
Software Report Card

"... a superior buy . . ."
Van Court Hare
in *Lifelines/The Software Magazine*

Don't waste another minute on a slow language processor. Order your BDS C Compiler today!

Complete Package (two 8" SSDD disks, 181-page manual): **\$150**
Free shipping on prepaid orders inside USA.
VISA/MC, COD's, rush orders accepted. Call for information on other disk formats.

BDS C is designed for use with CP/M-80 operating systems, version 2.2 or higher. It is not currently available for CP/M-86 or MS-DOS.

BD Software

BD Software, Inc.
P.O. Box 2368
Cambridge, MA 02238
(617) 576-3828

Circle no. 12 on reader service card.

NEW FEATURES

(Free update for our early customers!)

- Edit & Load multiple memory resident files.
- Complete 8087 assembler mnemonics.
- High level 8087 support. Full range transcendental (tan, sin, cos, arctan, logs and exponentials) Data type conversion and I/O formatting.
- High level interrupt support. Execute Forth words from within machine code primitives.
- 80186 Assembler extensions for Tandy 2000, etc.
- Video/Graphics interface for Data General Desktop Model 10

HS / FORTH

- Fully Optimized & Tested for:
IBM-PC IBM-XT IBM-JR
COMPAQ EAGLE-PC-2
TANDY 2000 CORONA
LEADING EDGE

(Identical version runs on almost all MSDOS compatibles!)

- Graphics & Text (including windowed scrolling)
- Music - foreground and background includes multi-tasking example
- Includes Forth-79 and Forth-83
- File and/or Screen interfaces
- Segment Management Support
- Full megabyte - programs or data
- Complete Assembler (interactive, easy to use & learn)
- Compare

BYTE Sieve Benchmark jan 83
HS/FORTH 47 sec BASIC 2000 sec
w/AUTO-OPT 9 sec Assembler 5 sec
other Forths (mostly 64k) 70-140 sec

FASTEST FORTH SYSTEM
AVAILABLE.

TWICE AS FAST AS OTHER
FULL MEGABYTE FORTHS!

(TEN TIMES FASTER WHEN USING AUTO-OPT!)

HS/FORTH, complete system only: \$250.



Visa Mastercard
Add \$10. shipping and handling

HARVARD SOFTWARES

P.O. Box 2579
Springfield, OH 45501
513/390-2087

Circle no. 44 on reader service card.

get lucky and have your part in three weeks. Probably more, though, since Shugart has given up on that drive."

They have? I called Shugart. Yes, a nice lady in Marketing told me, they were no longer selling the 860. She thought it might be because the market for 8-inch drives was so much smaller than that for 5-inch ones. No, it definitely was not because the 860 was a bad drive. They were still making the 800 model, and they'd support the 860 for several years. Repair parts? Certainly; just call our one and only official distributor in Los Angeles.

The Tandon 848

So I've got not just a busted drive but a busted *orphan* drive, see? I'm out at least three weeks and \$300 for a new head assembly, then I'll have a working orphan drive. Rubbish. I called Priority One Electronics. What about the Tandon? Sure, it's a fine drive, we have 'em in stock, and they are 100% compatible. Plus they're on sale.

The new Tandon came in only two weeks, but the Tandon manual didn't; it was back ordered. The drive has at least as many jumper options as the Shugart, but (of course) they had different names silk-screened onto the board. The piece of cardboard that was stuffed into the drive's gullet to keep the heads apart during shipping had a cryptic table of jumper options. Some of the factory settings it showed didn't match the reality of the board. Others it named turned out not to be jumpers at all but traces that could be cut or soldered.

I called Priority One repeatedly; I even talked them into reading me the Tandon manual over the phone. The guy who read it to me was not the world's best interpretive reader. I got the impression that I couldn't use True Ready and control the motor with Head-Load Timing at the same time. There were too many ambiguities.

Finally, the manual arrived. With tax it cost just under a dollar a page. You know, the one really good thing about the Shugart 860 drives is the manual. It is clear and well organized. The Tandon manual isn't. Its description of the jumper options is cryptic, ambiguous, and actually wrong in a few places. No wonder I'd been confused by a hasty reading over the phone!

Fortunately, the drive itself is nice.

Much better than the Shugart. I can't say how robust the head suspension is, since the head assembly is invisible and I have no intention of taking a screwdriver to that drive. But it's much quieter in operation. The Shugart (the remaining 860 is now drive B) makes a harsh buzz when it seeks and loud snapping noises as its door-lock solenoid operates. The Tandon seeks with an oily purr that's barely audible over the hum of a cooling fan.

The Shugart doesn't know where its head is when it powers up; until it is homed, it won't read reliably. The Tandon homes itself when power is applied, so it is ready to go on the first command.

The Tandon isn't really compatible, though. Oh, it responds to the same commands the same way—almost. But its power supply connector isn't the same, and the threaded holes for side mounting are in different places than the Shugart. The Shugart door handle opens left; the Tandon opens to the right. The Shugart will seek to, and write on, track 77 (the 78th track), but the Tandon will go not-ready if you try that. I know, because my disk format routine had an off-by-one error that didn't show up until I tried it on the Tandon.

"Anyway, that's how I lost a drive and some downtime and about \$600. I have my half-heights now, though, even if they are of different makes. Plus an extra drive I can cannibalize for spare parts—poor headless thing."

The Intern tossed the crumpled coffee cup toward a box of bloody listings. "Thanks for listening." He wandered off down the hall, a rumpled figure under the cold lights. The wind howled outside, and the rain streamed down the windows.

DDJ

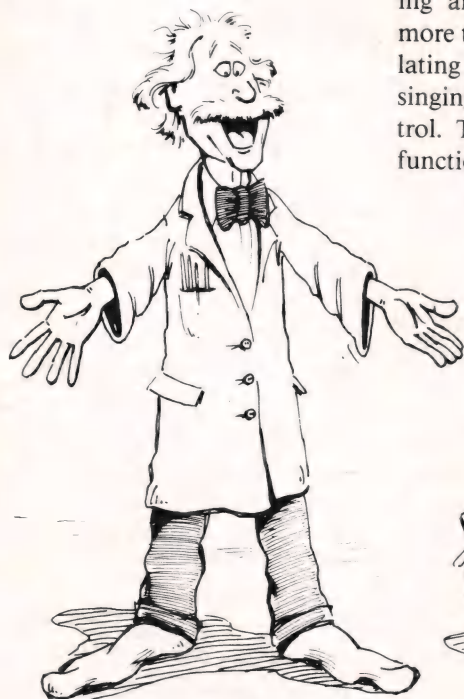
Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 190.

ROBOTICS AGE

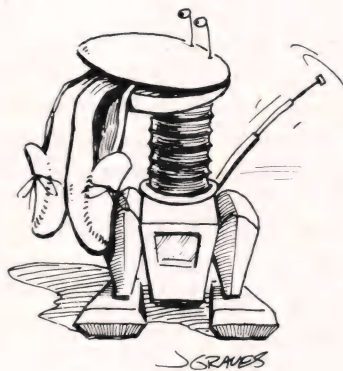
A Real-Time Experience

Learning how to program was the easy part. Now, what are you going to do with your knowledge? How about writing a check balancing program?



Maybe you could start developing that wonderful data base to keep track of everyone's birthdays. However, there are more interesting challenges.

Computers are very useful for tracking and filing information. There's more to computing than just manipulating data. There's walking, talking, singing, listening, touching, and control. The computer's most powerful function is control.



Real-time computer techniques can control factories and machinery, monitor your home environment and protect it from intruders, and operate little mechanical friends which will take out the garbage.

The near future will show us machines which respond to human voice controls, are capable of finding their own way around a house or factory floor, and are able to make their own decisions. *Robotics Age* teaches you to design and work with the practical real-time applications of state-of-the-art microcomputer technology. Robots are simply machines which respond to their environments and can act on their own. Computers make these machines possible.

After all, many people claim their computers are user friendly—but how can your computer be user friendly if it doesn't come when it's called?

It's time for you to experience *Robotics Age*. Explore the frontiers of microprocessor applications. Use the subscription form below to start the flow of vital technical information you need.

YES!

Sign me up **TODAY** for my personal subscription to *Robotics Age*,
The Journal of Intelligent Machines.

DD3

US Subscriptions

- | | |
|------------------------------------|------|
| <input type="checkbox"/> 12 issues | \$24 |
| <input type="checkbox"/> 24 issues | \$45 |
| <input type="checkbox"/> 36 issues | \$63 |

Canada & Mexico

- | | |
|------------------------------------|------|
| <input type="checkbox"/> 12 issues | \$28 |
| <input type="checkbox"/> 24 issues | \$53 |
| <input type="checkbox"/> 36 issues | \$75 |

Foreign

- | | |
|---|-------|
| <input type="checkbox"/> 12 issues (surface) | \$32 |
| <input type="checkbox"/> 12 issues (Air Mail) | \$68 |
| <input type="checkbox"/> 24 issues (surface) | \$61 |
| <input type="checkbox"/> 24 issues (Air Mail) | \$133 |
| <input type="checkbox"/> 36 issues (surface) | \$87 |
| <input type="checkbox"/> 36 issues (Air Mail) | \$195 |

Non US Subscription Rates:

Payable in US funds, drawn on a US bank. Subscription length will be adjusted downward on a pro-rata basis for any currency conversion charges. Foreign subscription orders may be paid in US dollars via MasterCard or VISA.



RETURN WITH PAYMENT TO:
Robotics Age, Box 358
Peterborough, NH 03458

Name _____

Company _____

Address _____

Town _____

State _____

Zip/Postal Code _____

Country _____

☐ Bill Me

Credit Card Information

☐ MasterCard

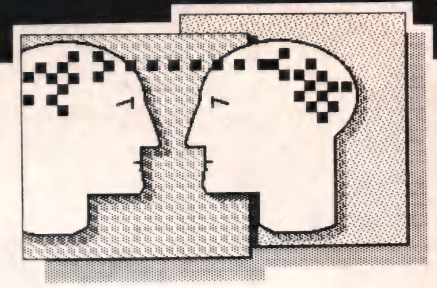
Card Number _____

☐ VISA

Expiration Date _____

Signature _____

Total amount Enclosed or Charged \$ _____



by Robert Blum

I've run a little late completing the subroutine to handle buffered disk I/O that I promised to run this month. In its place I want to talk about the early years of CP/M and how it became the most popular operating system for 8-bit computers. I also want to touch on the benefits of using disk sectors larger than 128 bytes. Both of these topics were requested by some nice folks that I recently met at a user group meeting. They thought a discussion without the bits and bytes, or at least minimizing the technical details, would be most helpful—especially for the hobbyist who is new to CP/M and isn't familiar as yet with assembly language programming.

In the Beginning

From its inception, CP/M was targeted for the 8008's successor and Intel's latest brainchild, the 8-bit 8080 microprocessor. While a software consultant for Intel, Dr. Gary Kildall had written the earliest versions of CP/M for his own experimental machine. His original development system included one of Shugart Associates first 8-inch disk drives, which had just come from equipment life testing and was about to take its last step. Fortunately, plenty of magic was left to finish CP/M. No one realized it at the time, but the combination of the 8080 CPU, CP/M, and two 8-inch disk drives was soon to become the standard of the then fledgling personal computer industry.

Within a few years, the price of the 8080 had dropped from its original \$400 – \$500 level to one that permitted a number of small companies (many working out of basements and garages) to begin shipping reasonably priced microcomputers, mostly as kits, to a growing audience of enthusiastic hobbyists. Although you couldn't do much more than programming for grins in 8K of memory, it became obvious very quick-

ly that one important element was missing: an operating system capable of supporting disk file management.

The decision of several companies not to develop their own operating system and the inability of others to deliver a suitable product were primarily responsible for CP/M's becoming today's pseudo-standard. Several of the first companies to adapt CP/M for their products made disk drive subsystems for S-100 bus-compatible machines. The prominence of the S-100 bus at this time focused even further emphasis on CP/M and detracted from other specialized research and development projects.

The average early microcomputer sported a 2 MHz clock rate and generally used less than the maximum 64K of main memory, placing obvious limitations on the resources available to CP/M. This limited environment helped sharpen the basic design goal of CP/M: to provide a straightforward, no nonsense approach to a single-user operating system.

Within just a few years, CP/M was offered on practically every 8080-compatible 8-bit computer system built and was being adapted or at least planned for every new system to come along. This nearly universal acceptance of CP/M heightened the need for enhancements to the disk interface portion of the BDOS to ease the adaptation process.

Version 2.2 of CP/M was brought out in the early '80s. Its most exciting feature was a new, flexible, table-driven generic disk interface and the ability to address larger capacity disk drives. Not only was the job of integration greatly simplified, but the flexibility of the new interface allowed many new storage options to be offered by the computer manufacturers. It soon became common to offer several different disk capacity options for the

same computer.

Within the last year, DRI has released its latest revision of CP/M called CP/M Plus. This version turns out to be a completely new system, structured for an environment that includes expanded memory of at least 96K. As expected, when given enough memory to perform all of its magic, CP/M Plus can improve the runtime of programs that make heavy use of the disk system.

For over 10 years, CP/M has led the way in 8-bit operating systems; it continues today to maintain its position of prominence by having more installed systems than any competitor. Since those early days, CP/M has gone through a number of upgrades but no change in philosophy. It remains the same rock-solid generic disk operating system that we have grown up with.

How It Operates

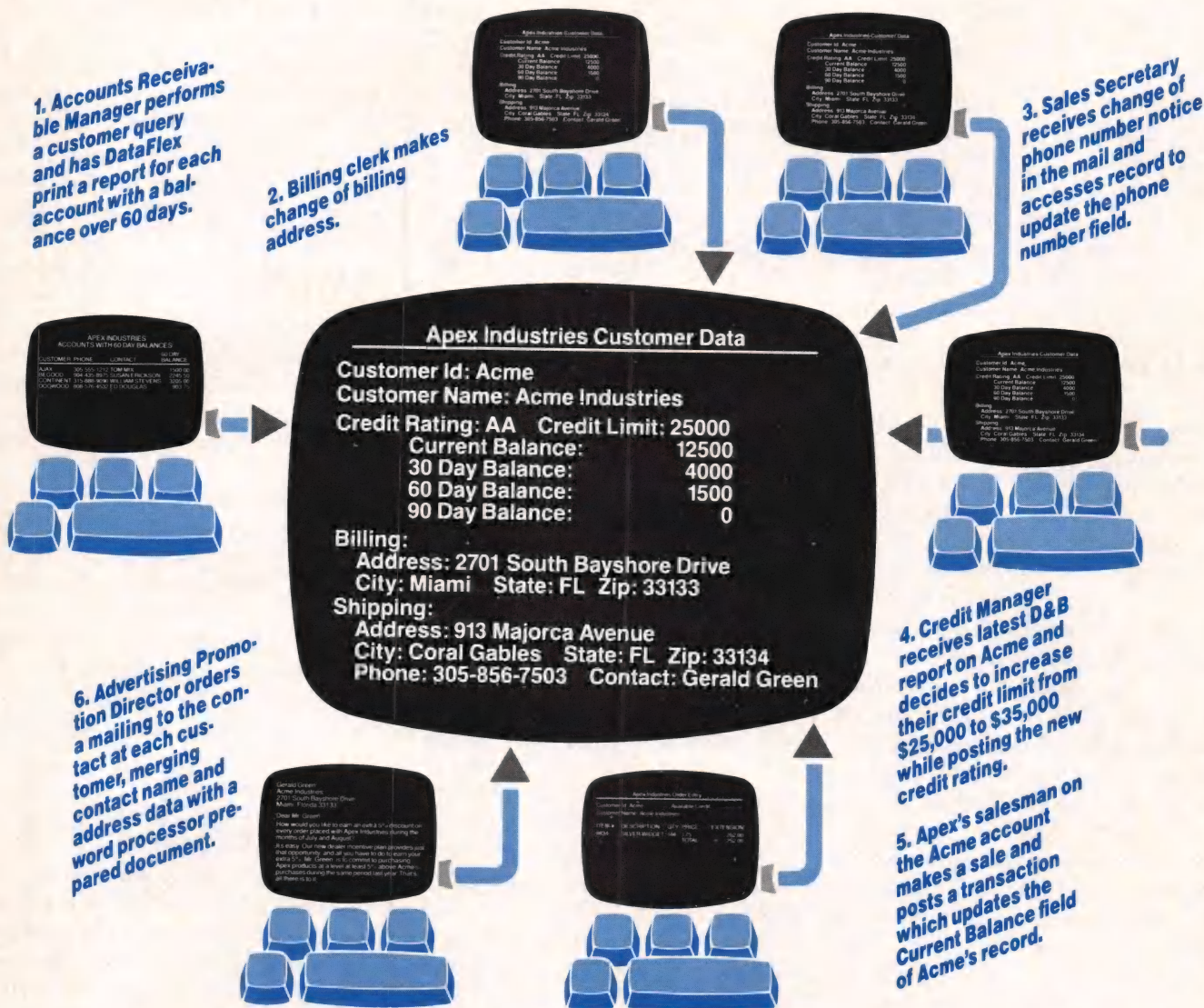
CP/M is simple-minded in its dealings with the host I/O system (BIOS). The rules are few; as long as they are religiously followed, the marriage will remain a peaceful one.

The portion of CP/M that intercedes between the computer's hardware and the BDOS is the BIOS. Contained in the BIOS are all the machine-dependent routines needed to interpret CP/M's language to that of the hardware system. One major task of the BIOS, and probably the most important, is to maintain peace and order over the disk system.

Most of the time during a disk I/O operation is spent waiting for the drive's mechanical apparatus to properly position itself for the data transfer. Consider for a moment the number of interrelated mechanical events that must happen in precise order to prepare for a single data transfer. First, the spindle motor is started and allowed to stabilize at a constant speed.

ALL AT ONCE!

AND NEVER A "LOCKED OUT" USER!



DataFlex is the only application development database which **automatically** gives you true multi-user capabilities. Other systems can lock you out of records or entire files for the full time they are being used by someone else. DataFlex, however, locks only the data being changed, and **only** during the micro-seconds it

takes to actually write it to the file! The updated record is then immediately available. The number of users who can access, and change, records at the same time is limited only by the number of terminals on your system or network. Call or write today for all the details on DataFlex... the true multi-user database.

DATA FLEX™

DATA ACCESS CORPORATION
 8525 SW 129 Terrace, Miami, FL 33156 (305) 238-0012
 Telex 469021 DATA ACCESS CI

See us at Comdex Booth 3349

Compatible with CP/M-80, MSDOS networks, MP/M-86, Novell Sharenet, PC-Net, DMS Hi-net, TurboDOS multi-user, Molecular N-Star, Televideo MmmOST, Action DPC/OS, IBM PC w/Corvus, OMNINET, 3Com EtherSeries and Micromation M/NET.

MSDOS is a trademark of Microsoft. CP/M and MP/M are trademarks of Digital Research.

Circle no. 29 on reader service card.

Next, the heads are loaded against the disk surface. Finally, the heads are stepped in or out to the proper track. At this point, to search for the requested sector and transfer it requires only a few additional milliseconds.

To complete the startup cycle and read one record on even the latest model disk drive can account for a delay of as much as one-third to one-half of a second. If this seems like a long time, it is—certainly long enough to be worthy of efforts to reduce it to a bare minimum. It's not hard to estimate how slowly a program would run if a complete startup cycle were necessary for each data record read. It stands to reason that as much data as possible should be transferred during each cycle.

The complexity of the BIOS is largely dependent on the hardware and whether the physical disk records are larger than CP/M's logical sector size of 128 bytes. If, for example, the disk being used is formatted with a physical sector size of 128 bytes, the BIOS has the fairly uncomplicated job of instructing the disk controller hardware where to put each data sector as directed by the BDOS. Depending on the hardware, this task can be as simple as loading several registers with the track and sector values and initiating the I/O operation by loading an instruction into the command register. If you are using one of the newer intelligent disk controllers that address the disk through data block numbers rather than by actual sector and track numbers, the job is slightly more complicated; some calculations are necessary but only a few.

The introduction of CP/M 2.x

brought with it the ability to easily use physical sector sizes larger than the standard logical sector size of 128 bytes. At this time, two new buzz words became prominent when referring to disk systems: single-density describes disk formats in which both the logical and the physical sector sizes are 128 bytes in length, and double-density is used to describe almost any disk format where more than one logical sector is contained in a physical sector. This new version of CP/M also made it easier for the system integrator to fine tune the disk system to the computer system.

There are two reasons for complicating what was once the very simple issue of disk I/O. Changing the physical disk sector size from 128 to 512 bytes, for example, should in theory increase the disk system's throughput by a factor of four because four times as much data is being moved in one operation: for each sector read from the disk, the next three reads should be satisfied directly from memory and at memory speed. In theory this may be true, but other factors generally prevent an increase of this magnitude. A goal of twice as fast is probably more in order.

Another benefit of using larger disk sector sizes is that you can store more data on the same size disk. Most of the recording area of a disk is used for storage of control information that separates the physical sectors and describes the contents of the data record. For example, each data record area is preceded by a series of sync bytes to assist the hardware in locking onto the recorded signal. Once in phase with the recorded signal, the data record descriptors, the data record, and a few error-checking bytes are read from the disk. To in-

crease the data content in each sector from 128 to 512, for example, would require reducing the number of sectors and slightly changing the amount of control information and its content.

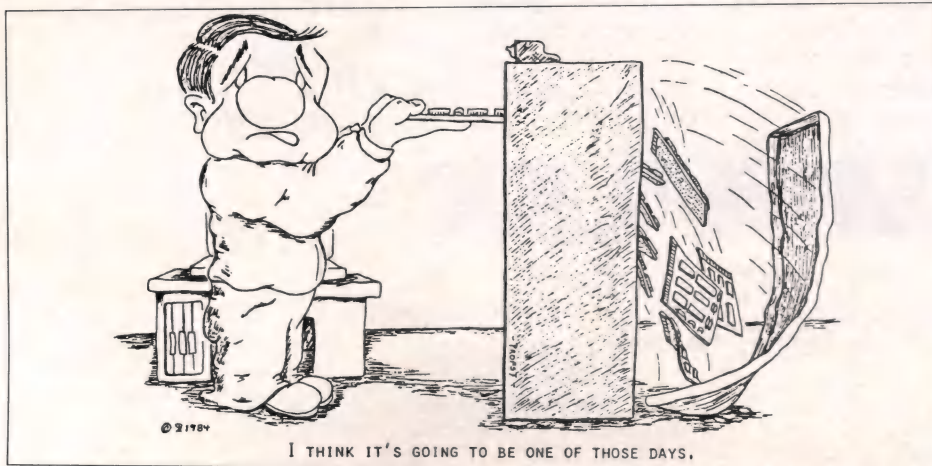
A perfect example of this is the double-density disk format used by Intercontinental Micro Systems (ICM). The standards published by Western Digital (ICM uses its 2793 disk controller chip) specify that, to ensure reliable operation, a maximum of 15–512 byte sectors be allocated to each track of an 8-inch disk. After exhaustive testing, ICM found that it could reliably format each track with 16–512 byte physical sectors. Formatting its double-density disks in this way permitted ICM to achieve the maximum possible disk capacity without using the extra memory required by an even larger sector size.

Using Large Sectors

To use physical sectors larger than 128 bytes requires that two routines be present in the BIOS. The first routine, blocking, maps CP/M's logical sector requests into the larger host buffer. The other routine, deblocking, performs the opposite operation of extracting the proper logical sector from the host buffer.

Imagine for a moment that a memory area 512 bytes in length has been set aside and divided into four 128-byte increments. Each of the 128-byte increments corresponds to a CP/M logical sector, and the entire 512-byte area corresponds to the host physical sector. When CP/M makes its first logical write request, the data is moved from the DMA address to the first slot in the host buffer. The second CP/M write request is placed into the second position of the host buffer, and so on until the fourth and last logical sector slot has been filled; the host buffer now must be emptied or data will be destroyed. From this example, we can see that using larger physical sectors allow the number of actual disk I/O requests to be reduced.

When CP/M makes a logical sector read or write request, the request is accompanied by the actual sector and track numbers. These two values are all that is needed to calculate exactly which host buffer contains the requested sector and where the sector is within



I THINK IT'S GOING TO BE ONE OF THOSE DAYS.

the buffer. If the request is a read, the host buffer is read into memory and the logical sector extracted. The buffer is kept intact in hopes that other sequential read requests will be made that can be satisfied directly from memory.

On the other hand, write requests are a little more difficult. Not only must the system place the record to be written into the proper slot in the host buffer, but any other data in the same buffer must not be disturbed. This usually requires that the host buffer first be read into memory, then modified by inserting the logical sector, and finally written back to disk. As before, the action of restoring the updated buffer on disk is held up in hopes that another sequential write request will be made—which, again, would be handled at memory-to-memory speed.

Everything that I have talked about works very well most of the time. A few situations, however, create severe buffer conflicts requiring two or three times more disk I/O activity than would be required on a single-density system. The cause of this problem is the bottleneck created by using only one memory-resident I/O buffer. For example, many traditional batch processing programs are written to input an old master file and write an updated one based on maintenance transactions input from a third file; each time a record is written to the output file, the output host buffer must first be read from disk, updated with the new logical record, then written back to disk. The additional I/O activity is due to the necessity of repeatedly writing the host buffer just updated back to disk to make room for the next incoming master file and transaction file record. This amounts to two extra I/O operations that wouldn't be required on a single-density system.

Several methods have been used to alleviate this problem. One allocates an extra memory buffer reserved exclusively for write operations. CP/M Plus also uses buffer memory, almost as much as you can give it, to buffer host sectors. Even at its worst, using larger physical sectors will give you a much more responsive disk system.

DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 191.

BRIEF

Program Editing is finally both: Intuitive and Powerful ... and configurable to suit your style

BRIEF lets you concentrate on programming by keeping the Editor "out of the way," while combining power and natural flow:

The New Standard. No longer does an Editor have to be "in your way" to provide full power. By combining power *with* natural flow, the new advanced BRIEF is in a class by itself.

BRIEF lets you concentrate on programming. Your thoughts flow smoothly, intuitively. 15 minutes is all you need to become fully productive. You can then do precisely what you want quickly, with minimum effort and without dull repetitions.

BRIEF adapts to your style. You can use BRIEF without modification, because it's distributed with an "ideal" configuration. Or you can make any change you want, add any feature of your own. Reconfigure the whole keyboard or just the Function Keys. Change the way the commands work or just the start-up defaults.

Availability: PC DOS-compatible systems with at least 192K and one floppy drive are required. Though your initial copy is protected, an unprotected version is available when you register BRIEF. Ask for special IBM AT or Tandy 2000 versions.

Pricing: Only \$195... with discounts for volume end-users. A demonstration version is available for only \$10 and can be available towards any Solution Systems purchase.

Win \$1,000 and substantial recognition for the Outstanding Practical BRIEF Macro. Other awards will also be given. Macros are fully programmable.

BRIEF'S PERFORMANCE IS NOT EQUALLED IN MICROS, MINIS AND MAINFRAMES

- | | |
|-------------------------------|--------------------------------|
| ■ Full UNDO (N Times) | ■ Windows (Tiled and "Pop Up") |
| ■ Edit Multiple Large Files | ■ Unlimited File Size |
| ■ True Automatic Indent for C | ■ Reconfigurable Keyboard |
| ■ Exit to DOS Inside BRIEF | ■ Online Help |
| ■ Uses All Available Memory | ■ Search for Complex Patterns |
| ■ Intuitive Commands | ■ Mnemonic Key Assignments |
| ■ Tutorial | ■ Horizontal Scrolling |
| ■ Repeat Keystroke Sequences | ■ Comprehensive Error Recovery |

PLUS a Complete, Powerful, Readable, Compiled MACRO Language

Try BRIEF. Use the Demo...
or the full product for 30 days.

Call or write us...
617-659-1571

**Solution
Systems™**

BRIEF is a trademark of UnderWare.
Solution Systems is a trademark of Solution Systems.

335-D Washington St., Norwell, MA 02061

Circle no. 93 on reader service card.

Varieties of Unix

An Introductory Guide to Microcomputer Unixes

Now available on computers from dozens of manufacturers ranging from Altos and Apple to Zentec and Zilog, including systems as small as the IBM PC and as powerful as the Cray supercomputer, Unix is well positioned to become a widely accepted standard operating system. According to a recent report issued by *Montgomery Securities*, "It is difficult to underestimate the importance of Unix to the computer industry In the business microcomputer world, Unix will simply become the industry's standard operating system. MSDOS will evolve to be Unix compatible and will be available as a subsystem under Unix."¹

By the time you acquire a general understanding of why Unix microcomputers are so attractive, you are likely to have discovered that there are several varieties of Unix, plus several "Unix-like" or "Unix look-alike" systems. What are the differences between these systems, and do the differences make any difference?

owner of the Unix trademark. True Unix systems are developed by purchasing a tape containing Unix source code from AT&T and enhancing and massaging the AT&T programs in accordance with the requirements of the intended environment. For example, a user-friendly front end menu system might be added to shield novices from the possibly intimidating terseness of the standard Bourne shell user interface.

Imitation Unix systems, such as Coherent, Cromix, Idris, QNX, uNETix, and UNOS, mimic Unix but do not contain the AT&T source code. Given the momentum now enjoyed by true Unix, the advantages, if any, of an imitation Unix ordinarily cannot compensate for the dangers involved in traveling along a nonstandard path. Of course, it all depends on your requirements: if you are doing nothing but word processing, and the only Unix feature you care about is the Unix tree-like hierarchical file structure, and are sure you won't want to use

There is no standard Unix. It's not even clear exactly what programs are defining features in any given implementation. Is it possible to extract a hypothetical standard from the union or intersection of existing implementations?

True versus Imitation Unixes

To qualify as true Unix, an operating system must be licensed by AT&T,

your computer for anything else in the future, then any number of operating systems imitating the Unix file structure could suffice. But if you care about the availability of a wide selection of software, want to stay flexible, and prefer to avoid unnecessary risks, you should stick with true Unix.

Cromemco's Cromix exemplifies some problems of look-alikes. Cromemco decided to imitate partly be-

Alan Walworth, Fortune Systems Corporation, 101 Twin Dolphin Drive, Redwood City, CA 94065.

Copyright © Alan Walworth 1984. All rights reserved.

cause Unix licenses were very expensive at the time (the price has since plummeted) and partly because major modifications, such as translating everything into Z80 assembly language and eliminating many capabilities, were needed to accommodate the limitations of an 8-bit 64K Z80 environment. Cromemco did a good job of fitting many Unix features into that environment: back in 1981, when it was first released, Cromix was impressive. In 1984, however, powerful, new, low-cost hardware makes it unnecessary to accept the constraints of an 8-bit processor with a 64K address space.

By using Z80 assembly language rather than the high level C language, Cromemco lost a major advantage of Unix: efficient portability to new hardware. Thus moving Cromix to the 68000 was a long and difficult task.

Except for software using Unix features that Cromix failed to implement, porting Unix software to the 68000 Cromix is feasible. The task of developing a Cromix version, however, typically ranks low (if it appears at all) on a Unix software developer's list of priorities; the result is that a person who opts for Cromix can make use of only a small fraction of the Unix software available to the owner of a true Unix system.² (In July 1984, Cromemco acknowledged the importance of standard Unix by unveiling new computers that run Unix System V.)

Because the Unix imitations, on the face of it, are not in the running, and because an analysis sufficiently detailed to provide insight into which ones might be worth considering in which special circumstances would have to be quite lengthy, the look-alikes will not receive further attention here.

A brief history of Unix will be presented to set the stage for a discussion of current microcomputer Unixes. Stanix, a hypothetical mainstream bare-bones version of Unix, then will be described in some detail, not only to provide general information about Unix but also to illustrate what is involved in analyzing a Unix system and to provide a paradigm useful as a starting point for describing various Unixes. Fortune Systems' FOR:PRO, Xenix, PC/IX, VENIX, System V, and 4.2 BSD will be examined next. Al-

though this survey includes most of the main microcomputer Unixes, some significant versions are regrettably omitted due to lack of sufficient information about them at the time of writing. Vendors annoyed by inadequate coverage here are invited to provide details about their Unix implementations to the author, so that a more comprehensive survey can be provided in a later article.³

A Brief History of Unix

Unix was developed in the early 1970s at Bell Labs by Ken Thompson and his associates. From the outset, the intent was to create a convenient and flexible environment for program development. Contrary to the tendency at that time to focus on maximizing the efficiency with which expensive hardware could be utilized, the developers of Unix concentrated their attention on efficient use of human resources (and, in particular, on efficient use of software developers' time). As hardware costs plummeted and software development costs rose, the wisdom of that approach became increasingly apparent.

Prior to Unix, operating systems had been written in assembly language, and Unix itself was originally written in assembly language for the DEC PDP-7 processor. In 1972, however, Unix was largely rewritten in C, a new language developed at Bell Labs by Dennis Ritchie, that combines low-level power with high-level convenience and portability. (A small part of the Unix code—about 5 percent—remains in assembly language for the sake of efficiency and because of the occasional need to use a hardware function not accessible via C.) Although programming the operating system in a high-level language makes its use of the hardware somewhat less efficient, that loss in efficiency is more than offset by the relative ease with which the code can be understood, maintained, enhanced, and ported to new hardware.

The distribution of Unix was constrained by federal restrictions on the AT&T monopoly's participation in the commercial marketplace; in 1973, however, Unix Version 5 (not to be confused with System V) was released to educational institutions and to some commercial organizations. PDP-11 minicomputers were widely used in

universities at that time, and Unix rapidly became popular as an operating system for those machines.

Version 6, released in 1975, was made available to commercial establishments as well as nonprofit organizations, but the price for the commercial market was very high, documentation was minimal, and support and maintenance were not provided.

Version 7, which appeared in 1978, contained enhancements including support of large files (up to one billion bytes), a standard I/O library, a more capable C compiler, an improved shell, and more sophisticated typesetting software. The University of California at Berkeley, Brian Kernighan's alma mater,⁴ ported Version 7 to its VAX minicomputers, and that was the configuration that was soon widely used at universities and other noncommercial institutions.

Bill Joy and his associates at the Berkeley Computer Science Department introduced a wide variety of enhancements, including the vi screen editor, the C shell, curses, and termcap. Berkeley enhancements were made available to the outside world in the 4.1 Berkeley Software Distribution (4.1 BSD). A more recent Berkeley version of Unix, known as 4.2 BSD, offers virtual memory, networking capabilities and faster file access.

System III, AT&T's first serious attempt to market Unix as a product, appeared in 1981. System III contained features from the Programmer's Workbench (PWB), which includes utilities such as the Source Code Control System (SCCS), Remote Job Entry (RJE), and nroff and troff.

System V, announced in 1983, is the version of Unix most widely used within the companies that once constituted AT&T and it is this version that the reconstituted AT&T would like to see accepted as a universal standard. System V contains many of the 4.1 BSD enhancements. It features Interprocess Communication (IPC), which employs named pipes, messages, shared memory, and semaphores. In January of 1984, AT&T announced a new release of System V, V.2.

Constant versus Variable Features

One of the reasons for Unix's populari-

ty is that it contains features now widely recognized to be essential for efficient operation of a multi-user system. The hierarchical file structure allows sensible organization of data. The file permission system protects files from unauthorized access or destruction. The password system limits use of the system to legitimate users. Convenient background execution improves user productivity by making it easy to complete time-consuming data processing tasks without interfering with ongoing work.

These basic features are explained in numerous introductory Unix texts, are fairly well known, and do not differ significantly from one version of Unix to another. Therefore, although they are extremely important, they will not be discussed further here. (For an explanation of such Unix fundamentals, see *Understanding Unix* by Groff and Weinberg, *The Unix Operating System* by Christian, or *The Unix Operating System* by Bourne.)

The following sections concentrate on two aspects of Unix that vary far more from version to version and are much more difficult to grasp: utilities and system calls. It is difficult to become familiar with these areas of Unix due to the vastness of the territory they cover. A typical Unix system has hundreds of utilities and dozens of system calls. Mastering many utilities requires a substantial effort because of their complexity, and understanding what certain system calls do (or even what a system call is) requires an understanding of internal computer operations more advanced than that enjoyed by many users.

The Utilities of Stanix

We will examine the utilities first because they are of primary interest to the majority of Unix users. The limited space available here makes it impossible to explain the utilities thoroughly, so we will simply list the most prevalent ones with a brief indication of what they do. The resulting checklist can serve as a starting point for detailed analysis of the utilities provided by various versions of Unix. Our immediate objective, however, is to use this list as a means of revealing, in general terms, both the contents of the Unix toolkit and the extent to which

acctom	searches and prints process accounting files
adb*	general purpose interactive debugger
admin	creates and administers Source Code Control System files (SCCS is a set of utilities for the administration of software development or document development projects. SCCS enables you to keep track of changes made to source code or text files, including the date of each change, the nature of the change, who made the change, etc.)
ar*	maintains archives and libraries
arcv	converts archives to a new format
as	assembler (all versions have assemblers, but they're not all called as)
at	executes a command at a specified future time
awk	a language for pattern scanning and processing
banner	prints large letters
basename*	removes "/" and "." extensions from a filename or path name
bc*	interactively processes arbitrary-precision mathematical expressions
bdiff	reports differences between two big files
bfs	scans big files
cal*	prints a calendar
calendar	reminder system
cat*	catenates (i.e., prints or lists) one or more files
cb*	C program beautifier
cc*	C compiler
cd*	changes the current directory
cdc	changes the delta commentary of a SCCS file
chgrp	assigns a file to a different user group
chmod*	changes file access permissions
chown*	changes the ownership of a file
chroot	changes the root directory for a process
clri	clears an inode
cmp*	compares two files
col*	takes reverse line feeds out of a file
comb	combines SCCS deltas
comm*	reports lines common and uncommon to two sorted files
cp*	copies a file or set of files
cpio	copies file archives in and out
cref	generates C program cross-reference listings
cron	executes commands contained in /etc/crontab at predesignated times
crypt*	encrypts or decrypts a file
csch	Berkeley's C shell
csplit	context file split
ctags	creates a function name index for a C or Fortran 77 source file
cu*	interactive system for calling another computer and transferring text files or for functioning as a terminal on the called system
cut	cuts out selected fields from each line of a file
cw	prepares constant width text for troff
date*	displays or sets the system date and time
dc*	interactively processes arbitrary-precision mathematical expressions
dcheck	checks the consistency of file system directories
dd*	converts and copies a file (e.g., to a non-Unix system)
delta	changes a SCCS file
deroff*	removes nroff, troff, tbl, and eqn constructs

Table I.
Stanix Utilities

df	displays statistics on disk usage
diff*	shows how two files differ
diff3*	reports differences among three files
diffmk	marks differences between files
dircmp	compares directories
du*	reports disk usage and file size statistics
dump	dumps selected parts of an object file
dumpdir	prints the names of files on a dump tape
echo*	displays arguments such as strings, filenames, shell variables, and command output
ed*	the old standard Unix line editor, useful in shell scripts
efl	extended Fortran language
egrep*	searches files for matches to a full regular expression
env	sets the environment for command execution
eqn	formats mathematical text for nroff or troff
ex	Berkeley's improved version of the ed editor
expr*	evaluates simple mathematical expressions and extracts strings
f77	Fortran 77 compiler
factor	factors a number
false*	returns a false value
fgrep*	searches files for matches to a fixed string
file*	reports the type of a file (executable code, text, etc.)
find*	locates files with specified properties
fsck	file system check with automatic repair option
get	gets a version of a SCCS file
getopt	parses command options
gets	suspends shell script processing to get user input
getty	sets the terminal mode and baud rate during startup
graph	draws a graph
grep*	searches for a text pattern in one or more files
head	displays the first lines of a file
help	provides helpful information about the SCCS
hyphen	finds hyphenated words
icheck	checks file system consistency
id	prints user and group IDs and names
init*	sets the environment for all user programs and allows users to log on to the system
join*	produces a join of two relational data base files
kill*	terminates designated processes
ld*	link editor
lex*	generates lexical analyzers
line	reads a line from standard input
lint*	reports possible problems in C programs
ln	makes a link to a file
login	logs the current user out and logs in a new user
logname	gets the current login name
look	finds lines in a sorted list
lorder	finds the ordering relation for an object library
lpq*	displays the printer queue
lpr*	spools a file for printing on a specified printer
lprm*	removes an item from the printer queue
ls*	displays a list of files in a directory
m4	a macro processor
mail*	sends mail to other specified users
make*	maintains sets of related program files
makekey	creates an encryption key
man	displays Unix Manual pages

Table I (cont)

various versions of Unix are the same at the utility level.

Utilities are simply programs that serve as useful tools. According to the Unix philosophy, each utility should perform a single task well. Power and productivity result from creatively combining the basic functions provided by the utilities. Some tasks performed by Unix utilities, such as copying or removing a file or logging in and out, are fundamental necessities for operating the computer. Thus, they are naturally regarded as part of the operating system (by definition an operating system is software that performs such fundamental functions). Traditionally, however, many other programs that are far less essential to operation of the computer have been regarded as part of the various Unix operating systems. To some extent, this is just a result of all programs that AT&T decides to provide on the Unix release tapes becoming automatically part of Unix. There is no other reason why Fortran, for example, is considered part of Unix and COBOL is not.

The Stanix utilities described in Table I (page 26) are standard in the sense that each appears in at least three of the following six major versions of Unix: Version 7, 4.1 BSD, System III, System V.2, FOR:PRO, and Xenix 3.0. This selection procedure, although somewhat arbitrary, produces a list of the mainstream utilities while winnowing out exotic commands like `rp6fmt`, `vpmc.u3b`, `300s`, and `4014`, which are of no importance to most Unix users. Of the 195 commands selected by this procedure, 92 are found in all six of the Unix versions; these are indicated in the table with asterisks.

Stanix System Calls

System calls are to a program what utilities are to the user: they are commands used by the program to tell the operating system to perform some basic action, such as opening a file for reading or creating a new process. Programs can also interact with the operating system by calling functions. The difference between the two methods is that the code for system calls is embedded in the operating system itself, whereas the code for a function must be obtained from outside the operating system.

As in the case of utilities, the list of

Stanix system calls in Table II (page 30) was derived by selecting those found in at least three of the following Unix systems: Version 7, 4.1 BSD, System III, System V.2, FOR:PRO, and Xenix 3.0. As before, asterisks indicate inclusion in all six of these Unixes.

Comparison of Six Major Versions of Unix to Stanix

Having presented Stanix, we can now examine the extent to which various versions of Unix differ from this standard. For each of the six major Unix versions—Version 7, 4.1 BSD, System III, System V.2, FOR:PRO, and Xenix—Table III (page 32) shows the number of Stanix and non-Stanix utilities contained in each version and the number of Stanix and non-Stanix system calls in each version.

The figures in Table III should not be used as a means of evaluating these versions of Unix. These figures are presented only as a method of conveying a general idea of the extent of these systems, the degree to which they overlap, and how much they differ. Although much effort has been devoted to arriving at accurate figures, inaccuracies are inevitable.

A major problem in compiling such statistics is that available listings of commands and system calls either contain omissions or, as in the case of complete sets of Unix manual pages, are too voluminous to digest in the time that was available for this study. The *Man Pages* tables of contents (the chief source used here for Version 7, 4.1 BSD, System III, and System V) fail to reference commands like *egrep* and *fgrep*, which are documented on the same man page as *grep*. Obvious omissions of this sort have been corrected, but others have probably slipped through.⁵ In particular, it is likely that the number of additional system calls shown in Table III for these systems is misleadingly low.

The number of additional utilities included is less meaningful than it might seem at first glance; some additional available commands were not counted because they are not considered *operating system* commands. This is especially true of FOR:PRO and Xenix. For example, communication utility programs available from AT&T for use on System V tend to become

mesg*	controls whether other users can write to your screen
mkdir*	makes a directory
mkfs*	constructs a file system on an unmounted device
mknod*	makes a device or special file
mm	prints documents using the mm macros
mmt	typesets documents, slides, and viewgraphs
more	displays a file one screen at a time
mount*	mounts a file system
mv	moves a file or set of files
ncheck*	displays a table of path names and inodes for a file system
newgrp*	changes your group identification
nice*	sets the priority at which a program should run
nl	line numbering filter
nm*	displays object file symbol names
nohup	executes a command that will ignore keyboard interrupts and will continue after the user logs out
nroff	a text formatter
od*	displays a file in requested formats such as octal, hex, and ASCII
pack	compresses and uncompresses files
passwd*	assigns or changes an account's password
paste	merges the same lines of several files or successive lines of a single file
pr*	paginates and adds an optional header
prof*	profiles program execution
prs	prints the deltas of a SCCS file
ps*	displays information about current processes
pstat	displays a table of system status information based on the state of the kernel
ptx	permuted index generator
pwd*	prints the working (current) directory name
quot	summarizes the file system ownership
ranlib	converts an archive file to a random library format that can be loaded efficiently
ratfor*	rational Fortran preprocessor
rc	invoked by init to run during startup and shutdown
regcmp	regular expression compiler
restor	restores the file system incrementally
rm*	removes a file or set of files
rmdel	removes a delta from a SCCS file
rmdir	removes a directory
sact	displays current SCCS editing activity
sccsdiff	compares two versions of a SCCS file
sdb	symbolic debugger
sdiff	side-by-side difference program
sed*	stream editor
setmnt	establishes a mnttab table
sh*	the standard Bourne shell
shutdown	the normal system shutdown program
size*	reports object file size statistics
sleep*	suspends shell execution for a specified interval
sort*	sorts lines of a file by specified fields
spell	checks spelling
spline	interpolates a smooth curve
split*	splits a file into pieces
strings	locates printable strings in a binary file
strip*	removes symbol table and relocation bits

Table I (cont)

struct	Fortran to RATFOR translator
stty*	displays and sets terminal characteristics
su*	temporarily changes your user ID so you can access restricted files and programs (The password must be given or the command will fail.)
sum*	calculates a checksum for a file to detect bad blocks
sync*	updates the super block on the hard disk and ensures that all disk writes have been completed
tabs	sets terminal tabs
tail*	displays the last lines of a file
tar*	tape archive utility
tbl	formats tables for nroff and troff
tc	typesetter simulator
tee*	copies standard input data, transferred through a pipe, to a specified file and to the standard output (A pipe passes the output of one command to the input of another.)
test*	tests to see if a condition exists or if a relational expression is true or false
time*	displays time statistics for the execution of a command
touch*	accesses a file without changing it to update its "date of latest access" field
tp	manipulates a tape archive
tr*	translates or filters specified characters
troff	formats text for typesetting
true*	returns a true value
tset	sets shell variables to accommodate a specific type of terminal
tsort*	sorts contents of a file based on a partial ordering of items in the file
tty*	displays the name of the current terminal's device file
umask	sets the permission default mask used in file creation
umount	unmounts a file system
uname	prints the name of the Unix system being used
unget	reverses a get on a SCCS file
uniq*	eliminates successive duplicate lines found in a file
units	measure conversion program
update	periodically updates the super block on the hard disk with sync to minimize data loss in the event of a system crash
uuclean	cleans up the uucp spool directory
uucp*	allows sophisticated automated communication between Unix systems
uux*	executes a command on a remote Unix system
val	determines if a file is a SCCS file with specified characteristics
vc	converts lines of input using specified arguments and control statements
vi	a sophisticated screen editor developed at Berkeley
vpr	Versatec printer spooler
wait*	stops interactive shell processing until all background processes are completed
wall*	broadcasts a message to all logged on users
wc*	reports the number of characters, words, lines, and pages in a text file
what	identifies SCCS files
who*	shows who is logged on
whodo	reveals who is doing what on the system
write*	allows interactive communication with other logged on users
xargs	constructs argument lists and executes commands
yacc*	yet another compiler compiler: a compiler creation system
yes	repeatedly displays a string or argument

Table I (cont)

part of the operating system by fiat, but when such a utility (e.g., a VT100 terminal emulator) is offered by Fortune, it is not considered part of FOR:PRO.

Finally, bear in mind that in some cases two versions of Unix have a utility (or system call) with the same name, but the functions performed, or the options available governing just what can be done with the utility or system call, are different.

With all their shortcomings, the numbers are still of interest. They show that the major Unix versions share a large body of commands that are essentially the same from one version to another. The body of shared utilities is sufficiently large that people familiar with one version will feel at home with another. They, however, may be annoyed at the lack of favorite utilities: programmers accustomed to Berkeley versions, for example, are likely to be disturbed when they find the C Shell (csh) is missing from the AT&T versions.

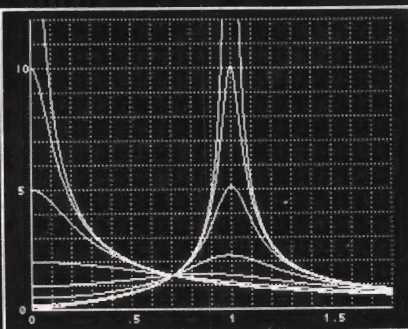
This analysis of the extent to which major Unixes overlap suggests two guidelines for the evaluation of any particular version of Unix: (1) if the version does not contain a healthy majority of the 195 Stanix utilities, it is, on the face of it, seriously incomplete, and (2) it is important to check to be sure that all the utilities you will need are available. Of course, novices who have no idea what utilities will be needed and who intend to avoid direct interaction with the operating system can apply the second guideline only by relying on expert assistance, the recommendations of experienced users, or the general reputation of the product to establish that the system will be able to perform as required.

Having gotten a feel for the extent to which these main versions of Unix differ from one another, let's turn to a consideration of the noteworthy features of various microcomputer Unixes. FOR:PRO will be discussed at some length to illustrate the sort of customization of Unix that is needed to create an effective and friendly microcomputer operating environment. We'll then look at other important microcomputer Unixes, including some that could not be included in the analysis above due to lack of data.

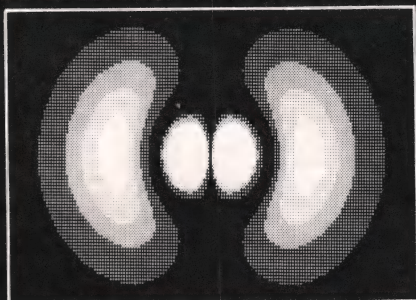
isys FORTH

for the Apple®][

Fixed point speed can rival that of floating point hardware. But the details have been a well kept secret—until now. The following graphs were generated by fixed point examples from the ISYS FORTH manual.



Parallel Resonance with Damping
BASIC 213 sec ISYS FORTH 27 sec



Hydrogen 3p Orbital Cross-section
BASIC 492 sec ISYS FORTH 39 sec

- Fast native code compilation. Sieve benchmark: 33 sec
- Floating Point—single precision with transcendental
- Graphics—turtle & cartesian with 70-column character set
- Double Precision including D*/
- DOS 3.3 Files read & written
- FORTH-83 with standard blocks
- Full-Screen Editor
- Formatter for word processing
- Macro Assembler
- Price: \$99, no extra charges

ILLYES SYSTEMS

PO Box 2516, Sta A
Champaign, IL 61820

Technical Information:
217/359-6039, mornings

For any Apple][model, 48K or larger.
Apple is a registered trademark of Apple Computer.

FOR:PRO

Fortune's FOR:PRO is Version 7 Unix with some System III software, 4.1 BSD enhancements, and Fortune enhancements. Thus, FOR:PRO is a variant of mainstream Berkeley Unix. Like System V, it contains many of the most desirable Berkeley enhancements.

In addition to Berkeley enhancements, FOR:PRO contains enhancements made by Fortune. These are of three kinds: bug fixes, changes for improved performance, and alterations needed to adapt Unix to a microcomputer environment. We will examine briefly the major changes of the last kind, beginning with alterations related to the Fortune computer's floppy disk drive.

The Unix source code provided on AT&T's distribution tapes is designed for use on a minicomputer that employs tape drives for backup and initial loading of programs. Although a tape

backup unit is available for the Fortune micro, the floppy disk drive included with the base system has to provide backup capability for users who don't have the tape streamer; the floppy drive is also normally used for new software installation. Thus, FOR:PRO needs driver software that can interface with the floppy drive.

More interestingly, it needs a version of the cp (copy) command that, unlike the standard Unix cp, is intelligent enough to pause and suitably prompt the user when a new floppy disk needs to be inserted during backup of a set of files too large to fit on a single disk. Similarly, when restoring a file system from a set of floppies, the Fortune cp needs to know enough to prompt for insertion of the next disk. A less essential Fortune enhancement to cp is the recursive option, which simplifies backup onto floppy disks by enabling a single command to be used to

access*	reports the accessibility of a file based on its permission modes and the real user ID of the user
acct*	initiates records for each system process in a file, or disables the record-keeping mechanism
alarm*	causes a signal to be sent to the current process after a specified number of seconds
brk	changes the amount of memory accessible by the current process
chdir*	changes the current working directory
chmod*	changes the access permission modes of a file
chown*	changes the owner and group of a file
chroot	changes the relative root directory for file identification
close*	closes a file
creat*	creates a new file
dup*	returns another file descriptor for a previously opened file so that the file can be accessed by two different file descriptors
exec*	executes an executable file
exit*	terminates a process
fcntl	provides control over open files
fork*	creates an identical twin (child) process
fstat	returns file status
getpid*	returns the process ID of the current process
getuid*	returns the real user ID of the current process
ioctl*	controls the operation of a terminal or other character device
kill*	sends a signal to a specified process, often resulting in termination of that process
link*	assigns an additional name to a previously existing file
lock	hastens execution by virtually preventing the current process from being swapped out of memory
lseek*	changes the position of the read/write pointer in a file
mknod*	creates a directory or device file
mount*	mounts or unmounts a file system

Table II.
Stanix System Calls

copy an entire file system, including the contents of subdirectories.

Unix systems normally require reconfiguration when hardware components such as new memory boards or I/O processors are added. In other words, the operating system must be told about the new hardware and reconstituted to take the new capabilities into account. Some Unix micros lack reconfiguration capabilities, making hardware additions impossible.⁶ On others, reconfiguration procedures require expertise that new users lack. Fortune avoids this difficulty with auto-configuration: when the Fortune system is powered up, it automatically determines what hardware is in the system and configures itself appropriately. For users with sufficient expertise, additional facilities for fine-tuning the system will optimize performance based on the nature of expected system usage.

A third major Fortune enhancement

is a menu shell that enables novices to use applications programs and administer the system without having any knowledge of Unix. Although the menu system is helpful for beginners, experienced users find it easier to work directly with Unix using either the Bourne shell or the C shell. As users learn more about Unix, they can easily switch back and forth between the Bourne and menu shells, letting the menu system assist with tasks they do not yet know how to perform with regular Unix commands.

As a rule, business microcomputer users are far less tolerant of operating system problems than engineers and computer scientists. For the latter sort of user, a system crash may be a minor annoyance; for the business user, it is more likely to be a major trauma. This is the case both because downtime and lost data can be very costly and because the business user typically has no idea what to do when something goes wrong.

nice*	sets the execution priority of the current process
open*	opens a file for reading, writing, or appending
pause*	suspends execution of the current process until a signal is received
pipe*	allows two processes to communicate by creating a mutually accessible data buffer
profil*	monitors or disables the user's program counter during the execution of a C program
ptrace*	traces and controls a child process
read*	reads data from a file
setpgid	sets the process group ID
setuid*	sets the effective user or group ID of the current process
signal*	intercepts signals for analysis, often preventing termination of the current process
stat*	reports the attributes of a file
stime*	sets the system date and time
sync*	writes all important memory data to disk
time*	returns total seconds since January 1, 1970
times*	returns system time consumption statistics for the current and child processes
ulimit	gets and sets user limits
umask*	sets the default mask used to establish access permissions for files
umount	unmounts a file system
uname	returns the name of the current Unix system
unlink	unlinks a filename from a file, and if it is the last filename or link to the file, removes the file from the file system
ustat	returns file system statistics
utime*	changes the record of the latest time at which a file was accessed and modified
wait*	suspends execution of the current process until a child process terminates
write*	writes data into a file

Table II (cont)

C

Software Development

PCDOS/MSDOS

Complete C Compiler

- Full C per K&R
- Inline 8087 or Assembler Floating Point, Auto Select of 8087
- Full 1Mb Addressing for Code or Data
- Transcendental Functions
- ROMable Code
- Register Variables
- Supports Inline Assembler Code

MSDOS 1.1/2.0

Library Support

- All functions from K&R
- All DOS 2.0 Functions
- Auto Select of 1.1 or 2.0
- Program Chaining Using Exec
- Environment Available to Main

c-window™

Symbolic Debugger

- Source Code Display
- Variable Display & Alteration Using C Expressions
- Automatic Commands
- Multiple Breakpoints by Function & Line Number

8088/8086 Assembler

- FAST — Up to 4 times Faster than IBM Assembler
- Standard Intel Mnemonics
- Compatible with MSDOS Linker
- Supports Full Memory Model

8088 Software Development Package

\$199⁰⁰

Includes: C Compiler/Library, c-window, and Assembler, plus Source Code for c-systems Print Utility

c-systems

P.O. Box 3253
Fullerton, CA 92634
714-637-5362

Two years of selling Unix systems to the business marketplace has given Fortune incentive to get the glitches out of its version of Unix. The result is that FOR:PRO is now an exceptionally robust Unix implementation. For the same reason, FOR:PRO now comes with a set of documentation oriented to the needs of inexperienced users.

Xenix

Xenix is Microsoft's adaptation of Unix System III. Like FOR:PRO, Xenix includes ex and vi; it does not include an office oriented word processor comparable to Fortune's Fortune:Word.⁷ For those wishing to adhere to the traditional Unix approach to word processing—an approach more suited to program documentation and writing scientific articles than to office automation—Xenix provides tbl for creating tables and eqn for representation of complicated mathematical expressions.

Also offered are the diction and style programs, which search for awkward expressions, provide a measure of how difficult your writing is to read, and so on. Xenix provides a novice-friendly menu system called the "Visual Shell." Because it's also available for MSDOS systems, this shell offers a means of providing a consistent user interface on all micros in an environment containing both Unix systems and IBM PCs.

In addition to uucp, Xenix includes Microsoft's micnet communications software, a less elaborate system that is somewhat easier to use than uucp. Unlike FOR:PRO, Xenix does not provide on-line documentation. However, a help facility is included in the visual shell. Xenix 3.0 provides utilities for reading and writing MSDOS files, a convenient feature for anyone who wants to run both MSDOS and Xenix on the same computer.

PC/IX

PC/IX is a fairly complete version of Unix System III, developed for the IBM PC by Interactive Systems Corp. of Santa Monica. PC/IX is said to be fast and to contain a good reconfiguration capability, but it is available only as a single user system and lacks some standard Unix utilities. The missing programs include fsck, pstat, ranlib, tar, and the C shell, csh. Although lpr

is not included, a sophisticated print spooling utility called print is supplied instead. INED, a good screen editor derived from the Rand e editor, is offered in place of vi.

Utilities are provided for moving files back and forth between PCDOS

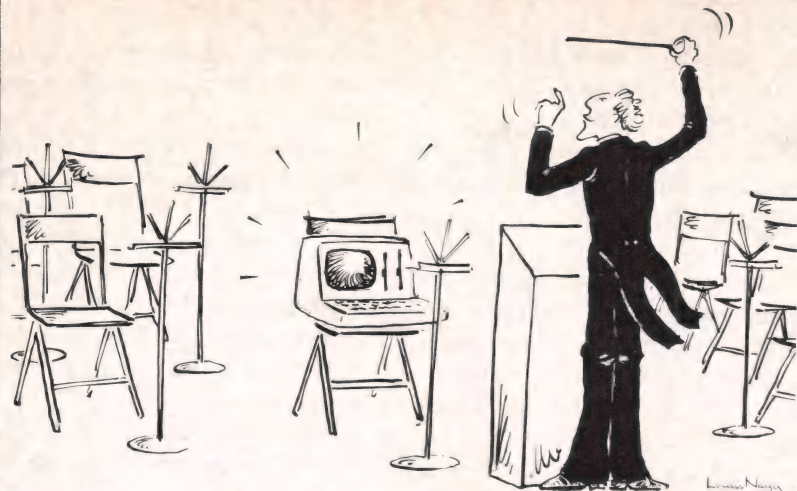
and PC/IX. No menu shell is offered. The troff supplied with PC/IX produces output for only a single phototypesetter, the Graphics Systems CAT. Compilers are available for C and Fortran 77; interpreters are included for BASIC and a version of SNOBOL.

Unix version	Version Seven	4.1 BSD	System III	System V	FOR:PRO	Xenix
Number of Stanix utilities included that are in all six versions	92	92	92	92	92	92
Number of additional Stanix utilities included (Total number of additional Stanix utilities is 103.)	34	51	74	72	87* (54)	79** (41)
Number of additional non-Stanix utilities included	16	105	79	140	57	49** (22)
Number of Stanix system calls included (Total in Stanix is 49.)	43	41	47	48	44	47** (46)
Number of additional system calls	6	15	1	10	20	23** (15)

* This is a projected number for early 1985. The number in parentheses reflects what is officially available at the time of writing (August 1984).

** This number is for the 3.0 release of Xenix. The number in parentheses reflects the pre-3.0 situation.

Table III.
Comparison of Six Unix Versions



Would you hire an entire band when all you need is one instrument? Of course not.

So why use a whole orchestra of computers when all you need is one to develop software for virtually any type of micro-processor?

The secret? Avocet's family of cross-assemblers. With Avocet cross-assemblers you can develop software for practically every kind of processor — *without having to switch to another development system along the way!*

Cross-Assemblers to Beat the Band!

Development Tools That Work

Avocet cross-assemblers are fast, reliable and user-proven in over 4 years of actual use. Ask NASA, IBM, Xerox or the hundreds of other organizations that use them. Every time you see a new micro-processor-based product, there's a good chance it was developed with Avocet cross-assemblers.

Avocet cross-assemblers are easy to use. They run on almost any personal computer and process assembly language for the most popular microprocessor families.

Your Computer Can Be A Complete Development System

Avocet has the tools you need to enter and assemble your software and finally cast it in EPROM:

VEDIT Text Editor makes source code entry a snap. Full-screen editing plus a TECO-like command mode for advanced tasks. Easy installation - INSTALL program supports over 40 terminals and personal computers. Customizable keyboard layout. CP/M-80, CP/M-86, MSDOS, PC DOS. \$150

EPROM Programmers let you program, verify, compare, read, display EPROMS but cost less because they communicate through your personal computer or terminal. No personality modules! On-board intelligence provides menu-based setup for 34 different EPROMS, EEPROMS and MPUs (40-pin devices require socket adaptors). Self-contained unit with internal power supply, RS-232 interface, Textool ZIF socket. Driver software (sold separately) gives you access to all programmer features through your computer, lets you download cross-assembler output files, copy EPROM to disk.

Model 7228 Advanced Programmer — Supports all PROM types listed. Super-fast "adaptive" programming algorithm programs 2764 in 1.1 minutes.

Model 7128 Standard Programmer — Lower-cost version of 7228. Supports all PROM types except "A" versions of 2764 and 27128. Standard programming algorithm programs 2764 in 6.8 minutes.

Avocet Cross-assembler	Target Microprocessor	CP/M-80	CP/M-86 IBM PC, MSDOS**
XASM04 <i>NEW</i>	6804	\$ 250.00	\$ 250.00
XASM05	6805	200.00	250.00
XASM09	6809	200.00	250.00
XASM18	1802/1805	200.00	250.00
XASM48	8048/8041	200.00	250.00
XASM51	8051	200.00	250.00
XASM65	6502/65C02	200.00	250.00
XASM68	6800/01, 6301	200.00	250.00
XASM75	NEC 7500	500.00	500.00
XASM85	8085	250.00	250.00
XASM400	COP400	300.00	300.00
XASMF8	F8/3870	300.00	300.00
XASMZ8	Z8	200.00	250.00
XASMZ80	Z80	250.00	250.00
XMAC682 <i>NEW</i>	68200	595.00	595.00
XMAC68K <i>NEW</i>	68000/68010	595.00	595.00

Model 7956 and 7956-SA Gang Programmers — Similar features to 7228, but program as many as 8 EPROMS at once. 7956-SA stand-alone version copies from a master EPROM. 7956 lab version has all features of stand-alone plus RS-232 interface.

EPROM: 2758, 2716, 2732, 2732A, 2764, 2764A, 27128, 27128A, 27256, 2508, 2516, 2532, 2564, 68764, 68766, 5133, 5143. **CMOS:** 27C16, 27C32, 27C64, MC6716. **EEPROM:** 5213, X2816A, 48016, I2816A, 5213H. **MPU (w/adaptor):** 8748, 8748H, 8749, 8749H, 8741, 8742, 8751, 8755.

7228	Advanced Programmer	\$ 549
7128	Standard Programmer	429
7956	Laboratory Gang Programmer	1099
7956-SA	Stand-Alone Gang Programmer	879
PDV	Driver Software	95
481	8748 Family Socket Adaptor	98
511	8751 Socket Adaptor	174
755	8755 Socket Adaptor	135
CABLE	RS-232 Cable (specify gender)	30

HEXTRAN Universal HEX File Converter — Convert assembler output to other formats for downloading to development systems and target boards. Also useful for examining object file, changing load addresses, extracting parts of files. Converts to and from Intel, Motorola, MOS, RCA, Fairchild, Tektronix, TI, Binary and HEX/ASCII Dump formats. For CP/M, CP/M-86, MSDOS, PC DOS. \$250

Ask about UNIX.

AVOCET'S SUPERB 68000 CROSS-ASSEMBLER — With exhaustive field testing completed, our 68000 assembler is available for immediate shipment. XMAC68K supports Motorola standard assembly language for the 68000 and 68010. Macros, cross-reference, structured assembly statements, instruction optimization and more. Linker and librarian included. Comprehensive, well-written manual. XMAC682 for MK68200 has similar features.

Call us toll-free for some straight talk about development systems.

1-800-448-8500

(in the U.S. Except Alaska and Hawaii)

VISA and Mastercard accepted. All popular disc formats now available - please specify. Prices do not include shipping and handling - call for exact quotes. OEM INQUIRIES INVITED.

*Trademark of Digital Research **Trademark of Microsoft

AVOCET SYSTEMS INC.™

DEPT. 1284-DDJ
804 SOUTH STATE STREET
DOVER, DELAWARE 19901
302-734-0151 TELEX 467210

VENIX

VenturCom's VENIX is an IBM PC implementation of Version 7 Unix with Berkeley enhancements, including vi and the C shell. Unlike PC/IX, VENIX can handle up to three simultaneous users. Awk, lex, and yacc are included, but lex is apparently incomplete.⁸ Troff is absent. Simple graphics functions are provided for tasks such as drawing lines and circles. On-line reference materials, available on full-blown Unix systems via the man command, are not included in VENIX. A BASIC interpreter is provided. Semaphores are available for interprocess communication, a step in the direction of System V's IPC capability. The Final Word is supplied as an optional word processing alternative. Of special interest to people dreaming of portable Unix systems is the plan (perhaps a reality by the time this is published) to offer a version of VENIX as an option on Data General's new 10-pound PC.

System V

In addition to bug fixes and performance enhancements, System V contains a variety of changes from System III. Improved file I/O performance may be noticeable due to a doubling of the block size to 1024 bytes.⁹ An improved scheme of file system updates reduces the risk of file corruption in the event of a system crash.

From a software developer's viewpoint, the most interesting improvement is System V's new Interprocess Communication (IPC) capability, which utilizes messages, shared memory, semaphores, and named pipes. The message capability allows a process to directly and efficiently communicate with another process. Shared memory is memory available to more than one process. Semaphores are used to coordinate access to shared memory by the multiple processes using it.¹⁰ Named pipes, like ordinary pipes, channel data from one process to another. The difference is that an ordinary pipe requires the two processes to be concurrent; with a named pipe, output from a current process can serve as input to a process that does not yet exist. An IPC remove command is available to clear out unwanted message queues and semaphore identifiers. A new file system checker, `dfsck`, allows multiple file

systems to be checked simultaneously.

The C programming environment has been altered in several respects, including a new Common Object File Format (COFF), enhancements to the math library, and `cflow`, a new program for flow analysis that generates module-calling hierarchy charts. An annoying deficiency of the System V C compiler is that it allows variable names to be only eight characters long (a limitation shared by many older C compilers). The improved symbolic debugger, `sdb`, can be used for both C and Fortran 77 programs.

The `-ms` macros, used by the `nroff` text formatter, have been eliminated.

4.2 BSD

4.2 BSD, the latest release of Unix from Berkeley, offers virtual memory, advanced networking capabilities based on "sockets," enhancements for fast file access, and the ability to use filenames much longer than the 14-character maximum on most Unixes. 4.2 BSD is the Unix of choice for engineers using relatively large and powerful systems such as superminis; it is less suitable for ordinary micros. Systems built around the 68000 processor, for example, lack the hardware required for effective virtual memory implementation (the 68010 overcomes this lack). Also, 4.2 BSD is relatively large, which interferes with efficient performance on a micro with limited memory. As micros become more powerful, these problems will become less significant.

The Future of Unix

AT&T is making a major effort to establish System V as the standard. But just as 4.2 BSD lacks some of System V's capabilities, System V lacks some important 4.2 BSD features. To some extent, we can expect the best features of both 4.2 BSD and System V to be incorporated into future Unix versions. The catch is that there is a limit to how many 4.2 BSD features can be grafted onto System V; addition of System V features to 4.2 BSD is far less problematic. Much depends on what IBM does, and strategists there must be tempted to thwart AT&T by promoting Unixes other than System V. (It is rumored that IBM is considering making 4.2 BSD available for use under VM.¹¹) So the extent to which System V will be

accepted as a standard remains in doubt.

Spending megabucks on advertising does not guarantee acceptance of a proposed standard. Micro Unix versions based on Version 7 and System III, such as Xenix, PC/IX, VENIX, and FOR:PRO, are in use at far more sites than System V. According to the *Supermicro* newsletter, "with the Xenix announcement for the AT, IBM has administered a possibly fatal blow to AT&T's expensive effort to establish System V as a standard."¹²

Jean Yates of Yates Ventures was quoted in September as saying, "Within 18 months AT&T will be conforming to the IBM standard—Xenix."¹³ Micro columnist John Dvorak recently suggested that "AT&T is not only losing key personnel but also may lose the System III vs. System V battle.... The failure of AT&T to add record locking and virtual memory to System V... may mean that AT&T will lose control of the direction Unix will take."¹⁴

It should be pointed out that AT&T is aware of System V's deficiencies and is working to overcome them. In a talk given in April 1984 at the European Unix User Group (EUUG) conference at the University of Nijmegen in Holland, Bell Labs' Larry Crume indicated that demand paging will be incorporated into AT&T's kernel as a configuration option in the near future.

The record-locking issue merits special attention because lack of record locking has often been mentioned as a weakness of Unix. To be sure, until recently there was no official Unix standard record-locking mechanism, and some Unix systems still don't offer record locking. But Unix systems aimed at the business marketplace, such as FOR:PRO, VENIX, and Xenix, must offer record locking because it's needed for efficient shared data base applications such as multi-user accounting systems.

The record-locking scheme developed by John Bass¹⁵ is the de facto standard and, as of the summer of 1984, is an official standard adopted by the `/usr/group` Standards Committee.¹⁶ In the original version developed for Onyx in 1981 and now in the public domain, the key system call is named `locking`. A more recent implementation calls the

UNIX FAMILY TREE

Bell Labs

Version 5 (1973)
PDP 11

Version 6 (1975)
PDP 11
First Public Release

AT & T

Version 6 PWB

Version 7 (1978)
PDP 11

Version 6
Mini Unix

32 V VAX

System III (1981)
VAX/PDP 11

System V (1983)
VAX

System V.2 (1984)
VAX

System V Micro Ports
(1984 - 1985)
M68000, Z8000, NS32000, Intel Z86

Berkeley

4.1 BSD
(1984) VAX

2.x BSD
PDP 11

4.2 BSD
VAX

Commercial Versions (1979 - 1984)
FOR:PRO, Xenix, PC/IX, UniPlus, Venix, etc.

Figure

key system call lockf. The lockf version is essentially the same as locking but provides enhancements such as the ability to test whether a region is locked without at the same time locking it. This record-locking scheme is effective and widely used, and AT&T has indicated it will cooperate with the /usr/group Standards Committee's adoption of it as a standard.

Choosing a Unix

Given the vast variety of Unix versions, which should you select if you are choosing a Unix system? Depending on your circumstances, you should consider not just what is contained in a given version of Unix but such aspects as:

- The likely enhancements in future releases of the version
- How fast the version runs
- How bug-free it is
- The quality of available documentation
- The quality of the hardware that the Unix version runs on
- Availability of support
- The range of software that runs on the version (in particular, whether all the application software you need is currently running in a reasonably well-debugged state on the system)
- Price

Don't decide you need System V just because AT&T has spent so much time and money advertising it. On the other hand, take all the talk about the failure of AT&T to make System V the standard with a grain of salt. As noted earlier, System V's deficiencies are being remedied. System V IPC capabilities provide an effective foundation for integrated application software; this inherent strength, as well as AT&T's marketing effort, will result in continu-

ing integration of System V features, if not true System V, into new microcomputer products. Pundits who proclaim that Xenix on the AT will blow System V out of the water ought to give some serious attention to Microsoft's efforts to produce a new version of Xenix based on System V. Although currently (August 1984) little if any commercially available software makes use of the new System V capabilities, a large quantity of high-quality System V software is likely to be on the market soon.

If the availability of the forthcoming System V software is important to you, bear in mind the possibility of avoiding current System V shortcomings by choosing a Unix that is not certified System V but that offers, or in future releases will offer, System V compatibility. Some microcomputer vendors who advertise System V in reality offer System V features grafted onto another version of Unix, rather than true System V. If it turns out that what is being offered is indeed real System V, be sure to check performance, reliability, and availability of adequate application software before you buy. Of course, these things ought to be checked before purchasing any other version of Unix, too. The special danger posed by AT&T's System V—and by IBM's products—is that massive marketing efforts, plus the magic of three familiar letters, may lead people to make purchasing decisions without being sufficiently careful to make sure that what they are getting satisfies their requirements.

Notes

- ¹ This report, titled "The Meaning of Unix," can be obtained by contacting Bill Shattuck at Montgomery

Securities (415) 627-2572.

- ² Note, however, that as of August 1984 System V Unix is an exceptional case; there is currently little application software for System V computers.

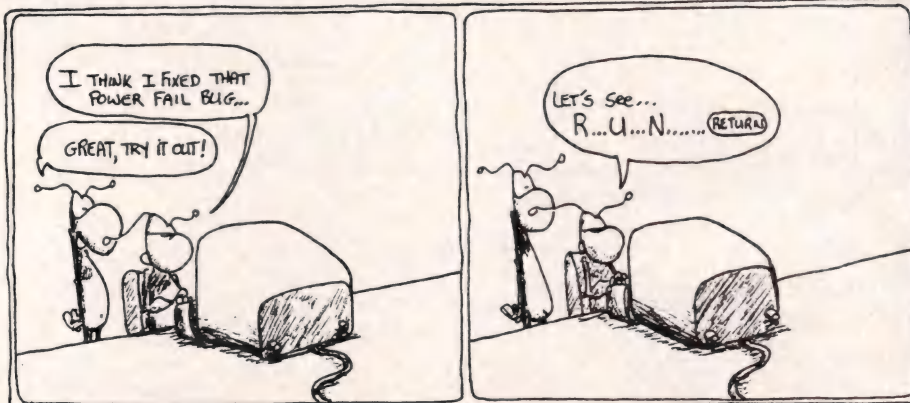
- ³ Information should be sent to Alan Walworth, 825 Clara, Palo Alto, CA 94303. Ideally, I would like many pages so that I can provide a detailed picture of a wide variety of Unix systems. Although ordinary sales literature is better than nothing, it is insufficient for in-depth analysis.

- ⁴ Brian Kernighan, coauthor with Dennis Ritchie of the key reference work *The C Programming Language*, was an early participant in Unix development at Bell Labs. Kernighan coined the term "Unix."

- ⁵ Note that, even if all these problems were taken care of, there would still be a similar problem with options, since you need to know all the options to grasp the full range of issuable commands. To see this point clearly consider that `egrep` and `fgrep` could have been implemented as options provided with `grep` (`grep -e` and `grep -f`). Such an implementation would have two fewer commands but no less functionality. So a comprehensive analysis of the capabilities of various Unix versions would require an investigation of just what command options, as well as just what commands, each provides.

- ⁶ Gene Dronek, "PCIX," *UNIX/WORLD*, March/April 1984, p. 37.

- ⁷ However, office-oriented word processing may exist for machines running Xenix. Microsoft and manufacturers of hardware on which Xenix runs should be able to provide



©R2/d92

information about what is available.

⁸ According to Mark S. Zachmann, lex "is missing one of its object libraries." This statement appears in his article "A Venerable UNIX," in *PC*, vol. 3, no. 11, June 12, 1984, pp. 246 - 248.

⁹ FOR:PRO also uses a 1024-byte block size. Using a large block size has both pros and cons: to take a simple example, if there are a thousand files, each 400 bytes long, using up a 1024-byte block for each one wastes a lot more disk space than using up a 512-byte block for each (roughly 1000 times 600, or 600,000 bytes wasted versus about 100,000 bytes). 4.2 BSD overcomes this drawback by using blocks 4096 bytes long for most files but arranging, when suitable, to let several small files share a single 4K block. The trade-off is that this and other file system optimization techniques add considerably to the size of 4.2 BSD.

¹⁰ For a good introductory discussion of semaphores, see *Operating System Design: The XINU Approach* by

Douglas Comer (Prentice Hall, 1984).

¹¹ Sol Libes, "News and Views," *Microsystems*, September 1984, p. 8.

¹² *Supermicro*, August 31, 1984 (published by ITOM International Co., P.O. Box 1415, Los Altos, CA 94022).

¹³ Jean Yates, *Fortune*, September 17, 1984, p. 70.

¹⁴ John Dvorak, *InfoWorld*, September 10, 1984, p. 96.

¹⁵ John Bass, a Unix expert who assisted with development of Fortune's FOR:PRO, is currently a consultant with DMS Design.

¹⁶ The /usr/group standards are published as a document titled *Proposed Standard: 1984 /usr/group Standard*, available from /usr/group, 4655 Old Ironsides Drive, Suite 200, Santa Clara, CA 95050. The *Reviewer's Guide to the Proposed /usr/group Standard*, which is available with the *Proposed Standard*, includes a listing showing which system calls and subroutines are provided in the /usr/group standard, System V, System III, Version 7, and

4.1 BSD. It also contains some useful background information, an explanation by John Bass of the lockf record-locking standard, and a report by D. Cragun and D. Kretsch of Bell Labs on the ways System V differs from the /usr/group standard. Cragun and Kretsch start off with a peculiar piece of reasoning that suggests AT&T is not wholly enthusiastic about the /usr/group standard: "The current membership of the /usr/group Standards Committee represent end users of Unix systems, developers of Unix systems, developers of systems that are based on or look like Unix systems, and applications developers. Therefore, the /usr/group standard is not a Unix system standard." /usr/group also publishes a *Unix Software Catalog*, a new edition of which is due out in early 1985; this could be a useful starting point for obtaining information on micro-computer Unixes omitted from this article.

DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 192.



UNIX

LET YOUR MODEM CONNECT YOU TO...

THE SOLUTION™

***EXPANSIVE SOFTWARE DEVELOPMENT FACILITIES.** Language and Operating System design.

- **LANGUAGES INCLUDE:** C, Fortran 77, RATFOR, COBOL, SNOBOL, BS, Assembler + LISP.
- **USENET Bulletin Board System**—800 + networked international UNIX sites, 190 + categories, 300 + new articles per day.
- **Inter/intra system mail + communications.**
- **UNIFY:** for professional data-base application development.
- **UNIX & System enhancements** from U.C. Berkeley and Kormeyer Electronic Design Inc.
- **Online UNIX manuals + Expert consultation** available.
- **Just a local modem call** from 300 + cities nationwide via Telenet.
- **Complete photo typesetting service.**
- **FAST and LOW COST,** low as \$8.95 hr. connect + .03 cpu sec.
- **\$24.95 = 1 hr. FREE system time,** SOLUTION News subscription, BYTE BOOK: Introducing The UNIX System, 556 pp. *UNIX is a trademark of Bell Labs.

Kormeyer
ELECTRONIC DESIGN, INC.

5701 Prescott Avenue
Lincoln, NE 68506-5155
402/483-2238 10a-7p Central

Payment via VISA or MasterCard

Circle no. 54 on reader service card.

Unix Device Drivers

by John L. Bass

Some Unix systems currently being shipped use disk queuing algorithms that optimize throughput. Their performance looks less attractive when you realize that the best-throughput algorithms yield the worst values for response time and fairness.

This article is an introduction to the Unix I/O subsystem and Unix device drivers. The concepts herein cover a wide range of Unix operating systems as well as a number of non-Unix systems. Unix Version 7 drivers are the baseline for the information in this discussion. Device drivers in both earlier and later versions of Unix are similar in concept to these Version 7 drivers but have different system interfaces, queuing structures, and control flow.

For more information on Unix, see *The Bell System Technical Journal*, Part 2, July – August 1978, vol. 57, no. 6. See also “The Unix I/O System” by Dennis M. Ritchie, found in the *Unix Programmer's Manual*, seventh edition, vol. 2B, and in other documents for various releases of Unix.

A Unix I/O Subsystem Overview

Before we look at Unix drivers, we need to take an overview of the Unix kernel to understand the overall system control flow. Shown in Figure 1 (page 40) is the block structure of the Unix I/O system—from a functional view. Each rectangular block in the diagram implements a certain functionality in a modular fashion. Communication between blocks is accomplished via subroutine calls and data structures passed as arguments in those calls. Each rectangular block with rounded corners represents a switch table structure used to select one of several similar functions.

The System Call Interface and Switch

System and application processes interface to the I/O subsystem via the library subroutines `open`, `close`, `read`, `write`, `ioctl`, and `lseek`. Each of these library subroutines copies its arguments according to system call conventions, sets the system call number, and then causes a hardware/software call to the system call manager. This procedure varies from system to system.

The system call manager copies the arguments out of the process' memory and into the kernel's memory. It then uses the system call number to index into the system call switch table (a jump table) to find the proper kernel subroutine to call. The arguments are assembled for the routine, and the routine is called.

When the routine returns, the system call manager copies the return status into the process' memory and resumes execution of the process.

John Bass, 6946 Blue Hill Drive, San Jose, CA 95129.



The File I/O Subsystem

The file I/O subsystem has entry points in the system call switch table for each file I/O system call. The following are the basic entry points used by applications:

Open: scans the directory structure to locate the file named by the application. The inode for the file is brought into memory for use by the other system calls. The inode describes all of the file type, ownership, and allocation information for a file.

If the inode describes a character or block special file, then the driver must be passed the open call. The major device number in the inode is used as an index into either the block or character device driver switch table to obtain the driver's open routine address. If no driver open routine address is found in the switch table, then the driver does not require an open call. If an address is found, then the open routine is called.

If an error occurred, then an error status is returned; otherwise, a file descriptor is returned to the application process.

Close: If the inode is for either a character or block special file and no other process has the inode open, then the close routine (if any) in the switch table is found and the driver's close routine is called.

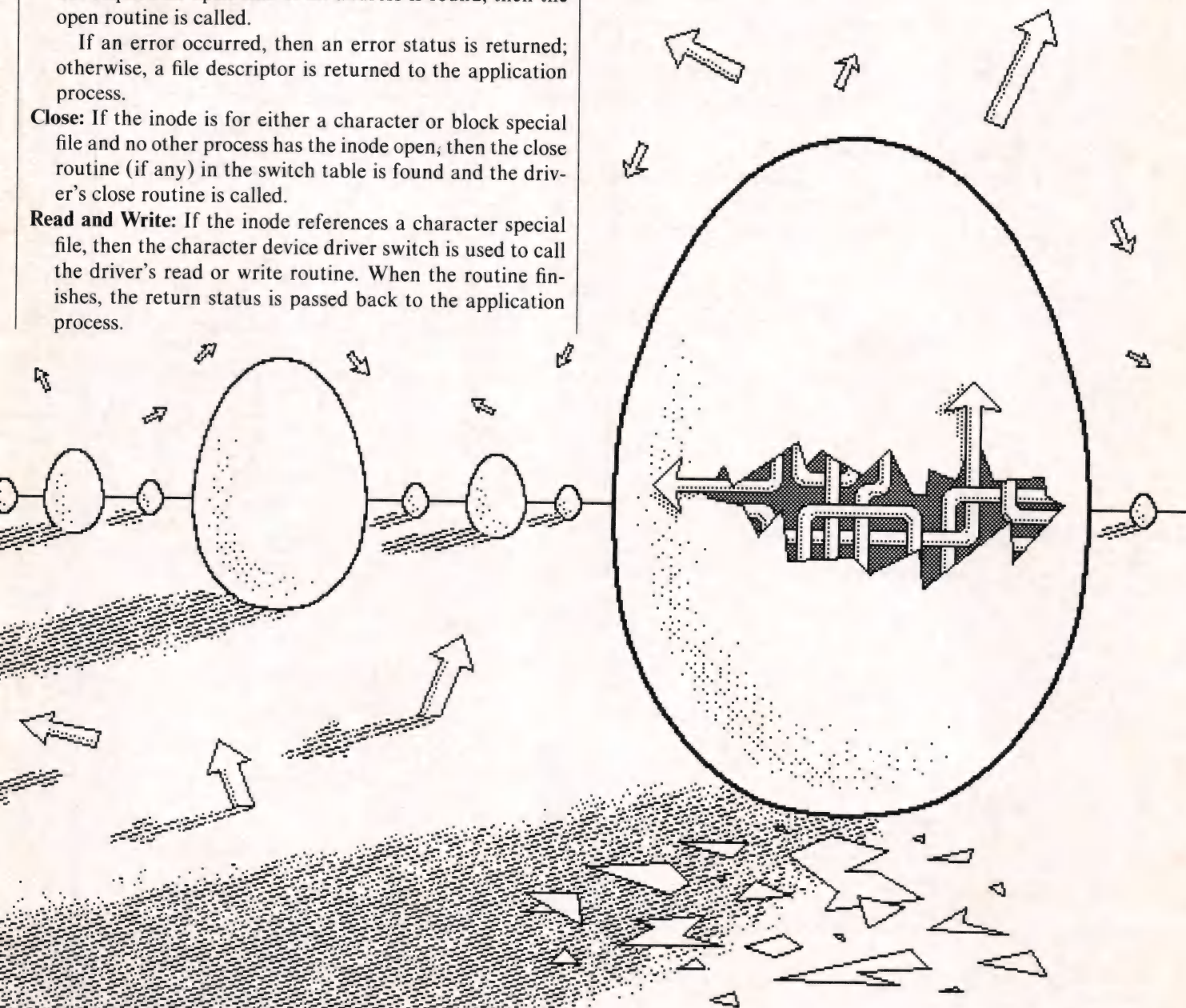
Read and Write: If the inode references a character special file, then the character device driver switch is used to call the driver's read or write routine. When the routine finishes, the return status is passed back to the application process.

If the inode references a block special file, then the current file pointer offset is translated into a logical block number for the device.

If the inode references a file or directory, the current file pointer offset is translated into a logical block in the file. The allocation information for the file is then used to locate the logical block number on the device.

The block I/O subsystem is then requested to return a system buffer with the device data. If it is a read system call, then the data in the buffer is copied into the applications buffer. If it is a write system call, then the data in the applications buffer is copied into the system buffer, and the buffer is flagged to be written back to the device.

This process is repeated for as much data as was requested by the applications system call.



Lseek: This system call does not cause any I/O action to be performed. It simply changes the current file pointer offset used by the read and write routines.

Ioctl: If the inode does not reference a character special file, then an error status is returned; otherwise, the ioctl routine in the device driver is called.

The Block I/O Subsystem

The block I/O subsystem maintains an in-memory buffer pool that forms a simple FIFO cache of blocks read or written. Requested blocks are returned immediately when they are found in the pool. If these blocks are not found in the pool, then the oldest buffer is freed, allocated for this device/block, and then passed to the proper device driver strategy routine.

The Process and Swap Scheduler

The system swapper uses a special buffer descriptor to build a swap in or swap out I/O request. The swapper then calls the device driver's strategy routine with the buffer descriptor to process the request. Normally, large swaps are done in a number of 10K to 60K chunks to minimize hogging of the device.

The Block Device Driver Switch

The switch table is an array of structures indexed by the block major device number. This structure contains the addresses for the device driver's Bopen, Bclose, and Bstrategy

subroutines. It also contains the address of the device's I/O queue for use by system metrics. Device drivers are not required to have a Bopen or Bclose routine. When they do not, the entry point for a null subroutine is placed in the table for the missing subroutine.

Each block device driver will have one or more entries in this table. The entries are created during system generation or, on some systems, during autoconfiguration when the system is started up.

The Character Device Driver Switch

The switch table is an array of structures indexed by character major device number. This structure contains the addresses for the device driver's Copen, Cclose, Cread, Cwrite, Cioctl, and Cstop subroutines. Also often included is a pointer to the first TTY structure controlled by the driver for use with system metrics. Device drivers are not required to have a Copen, Cclose, Cioctl, or Cstop subroutine. As with the block device, the address of a null routine is placed in the table. For a device that does not have a read routine (write only—e.g., something like a printer) or does not have a write routine (read only device—a paper tape reader), it is advisable to cause the missing function to return an error. For these devices, the table contains the address of a routine that returns an error.

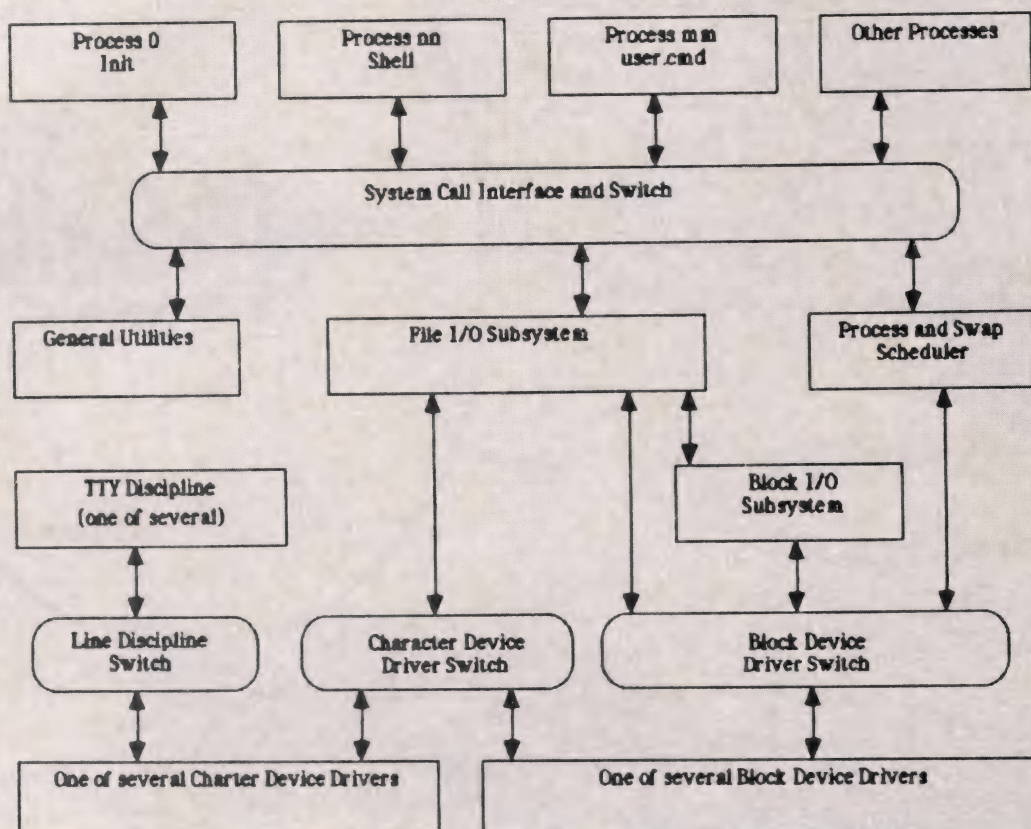


Figure 1
UNIX Functional Structure

The Block Device Driver

A generic disk driver represents the general form of most block device drivers. The structure of a disk driver is given in Figure 2 (page 41).

The block device drivers are used mainly by the file I/O subsystem and block I/O subsystem to provide buffered access to file system devices. In practical terms, these "devices" can be disk drive devices of all kinds, disk emulators (for instance, a RAM disk), and sometimes a network-provided remote disk service.

Most block device drivers also have a simple character driver interface as well. This interface allows direct multi-block transfer to and from the device without using system buffers. This process is mainly used by file system maintenance utilities. The character driver interface also allows the driver to use ioctl calls for special applications (for example, a Format Diskette command).

The Character Device Driver

Character device drivers are used to interface all types of

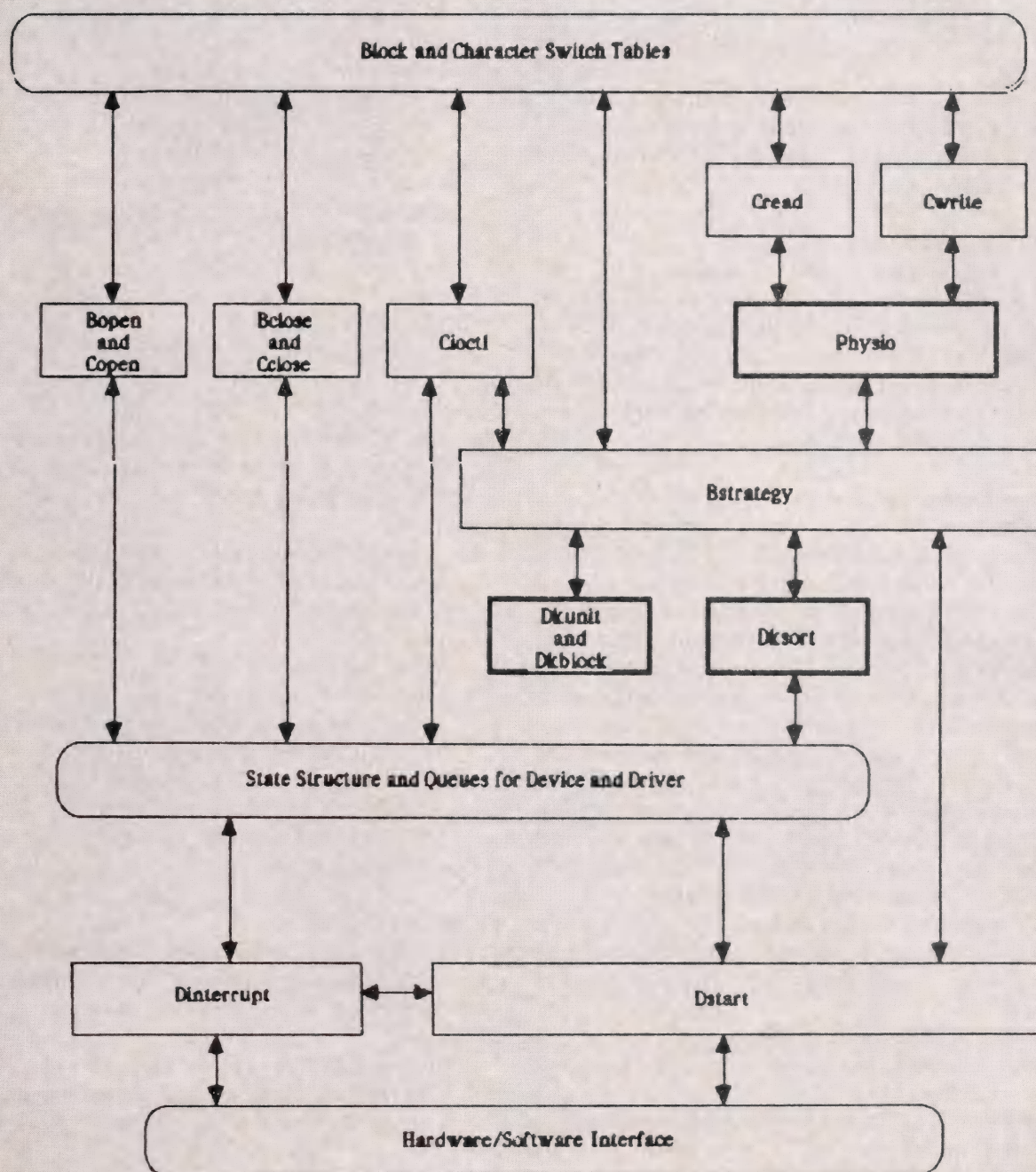


Figure 2
Disk Device Driver

devices to Unix. The driver interfaces the external device to open, read, write, and close system calls to maintain the device-independent I/O model.

An important specialized form of character device driver exists for communication devices that interface terminals to the system. All the special processing to handle echo, character erase, line erase, end of file, xon/xoff flow control, and signal generation is contained in the default tty line discipline. The line discipline switch is an interface to link the default tty line discipline into the terminal driver as well as provide for alternate tty protocols and special tty handling.

Much of the difference between Version 7, Berkeley releases, System 3, and System 5 revolves around changes and improvements in the tty line disciplines. There are significant differences between the implementations for each of these releases.

Figure 3 (page 46) shows the basic outline for a simple terminal driver. This driver completely defaults to using only the default tty line discipline. In practice, a fully implemented driver is much more complex.

The Block Device Driver

This section will discuss the generic disk driver in more detail (see Figure 2). Simple algorithms will be presented for each major routine found in the basic disk driver. The outlines provided here are simplified to some extent. The details vary slightly between various versions of Unix. If you have access to a driver source for your system, consult those listings for more details.

Subroutines Bopen and Copen

The basic function of an open routine is to provide initialization for the controller and drive. Both the block and character open routines may be active at the same time if a sleep is done during initialization of the controller or drive. Likewise, it is possible to have a concurrent close on one interface and an open on the other. Thus, if either open or close sleeps on some I/O, it may be necessary to set some semaphore to prevent race conditions during opens and closes.

There are separate open routines in both the block and character interfaces. To be certain that no minor device is open, the routine must check that no block minor device is open *and* that no character minor device is open. To forget this is a common mistake: it results in the controller/driver getting initialized while active I/O is in progress.

The open routine is called for each open system call for the minor device. The kernel should call open for the root, pipe and swap devices . . . but this is sometimes overlooked or difficult to do.

```
Begin subroutine open(minor_device)
  While open_close_lock set, sleep on open_close_lock
  Set open_close_lock
  If no minor device on this controller is open, then
    Initialize the controller
    Mark controller as ready
  End if
  If no minor device on this drive is open, then
    Initialize the drive
    Mark drive as ready
  End if
```

```
Mark minor_device as open
Free open_close_lock
End subroutine open
```

Subroutines Bclose and Cclose

This is simply the reverse problem from open with the additional task of flushing and waiting for outstanding buffered writes.

```
Begin subroutine close(minor_device)
  While open_close_lock set, sleep on open_close_lock
  Set open_close_lock
  If any buffered writes for this minor_device, then
    Flag and queue all buffers found
    Wait for minor_devices queue to become empty
  End if
  Clear minor_device open flag
  If no minor device on this drive open, then
    Take drive off line (if required)
    Release drive from controller
  End if
  If no minor device on this controller open, then
    Release controller
    Flag controller as idle
  End if
  Free open_close_lock
End subroutine close
```

Subroutine Cioctl

Most disk drivers do not have an ioctl interface. In those that do, the most common use of the interface is to provide a means for formatting floppy disks (or other removable media).

```
Begin subroutine ciectl(minor_device, cmd)
  If user is not root or cmd is not format, then
    Return error status
  End if
  While raw_buffer is busy, sleep
  Mark raw_buffer as busy
  Set up command parameters in raw_buffer
  Set command type to format
  Call strategy to queue I/O request
  While raw_buffer is not done, sleep
  Mark raw_buffer as free
End subroutine ciectl
```

Subroutine Cread

The normal raw disk read function simply calls physio to validate the user's request, acquire the buffer, initialize the buffer parameters, queue the request, and monitor the I/O completion.

```
Begin subroutine cread(minor_device)
  Call physio to build raw read request for strategy
End subroutine cread
```

Subroutine Cwrite

The normal raw write request is just like the read request.

```
Begin subroutine cwrite
  Call physio to build raw write request for strategy
End subroutine cwrite
```


FORGET

EVERYTHING YOU THOUGHT YOU KNEW ABOUT PROGRAMMING IN BASIC INTRODUCING

*Better*TM BASIC

*"It combines the best points of interpreted
BASIC, Pascal, Forth, & Assembler..."*

*...It's one of the few pieces of software I'd spend my
own money on."*

Susan Glinert-Cole
Technical Editor,
PC Tech Journal

Speed. Sieve of Erastosthenes Benchmark:

- BetterBASIC: 31.9 seconds.
- IBM PC BASIC: 191.1 seconds.

- ☐ Extensibility (Make your own BASIC!!)
- ☐ Program Block Structures.
- ☐ User defined Procedures and Functions.
- ☐ Local and Global Variables.
- ☐ Shared Variables.
- ☐ Recursion.
- ☐ Argument type validation.
- ☐ Optional arguments.
- ☐ Arguments passed by-value or by-address.
- ☐ Separately compiled program Modules.
- ☐ Simple interface to Assembly Language Procedures.
- ☐ Support for OEM hardware through extensibility.
- ☐ Useful set of Data Types:
 - Byte, Integer.
 - Real (Variable precision BCD) Ideal for business math.
 - String (up to 32768 characters).
 - Record Variables & Structures.
 - N-dimensional Arrays of any type.
 - Arrays of Arrays.
 - Pointer (of any type).

Support of large memory (to 640K). Built-in WINDOWS support!

- ☐ Interactive programming language based on an incremental compiler.
- ☐ Syntax checked immediately on entry, with concise error reporting.
- ☐ Built-in Screen Editor allows on-line editing.
- ☐ Built-in Linker for separately compiled program Modules.
- ☐ Built-in Cross Reference Lister
- ☐ Comprehensive 580 page manual.
- ☐ 8087 MATH support.
- ☐ IBM PC, IBM PC/XT or compatible.
- ☐ PC/DOS 1.1, 2.0, 2.1.



Summit Software Technology^{Inc.}

P.O. BOX 99, BABSON PARK, WELLESLEY, MA 02157

BetterBASIC is a trademark of Summit Software Technology, Inc. IBM PC, IBM PC/XT and PC-DOS are trademarks of International Business Machines Corp. MS-DOS is a trademark of Microsoft Corp.

We are so sure you will like BetterBASIC, we will give you a 30-day money-back guarantee. Order BetterBASIC now!
PRICE \$199.00
8087 MODULE \$99.00

Not convinced? Then try the BetterBASIC Sample and you will find that BetterBASIC is truly a major breakthrough in computer programming.
RUNTIME SYSTEM \$250.00
SAMPLE DISK \$10.00

You can find BetterBASIC by calling us at (617) 235-0729

Master Charge, Visa, P.O., Checks, Money Orders, and C.O.D. accepted.

System Function Physio

The system function physio does all the normal handling to set up a dma transfer to/from an applications buffer. This includes validation of the addresses and lengths requested by the application. The physical addresses are calculated and plugged into the provided buffer header. Physio then calls the provided strategy routine and waits for I/O to be completed. Any error processing is done, and the resulting status is returned to the application.

Subroutine Bstrategy

The strategy routine's job is to take the requested buffer header, map it to the correct device and physical block, and enter it into the queue.

```
Begin subroutine strategy(buf)
  If the requested I/O is outside this partition, then
    Set error flags
    Mark buffer as done
    Return to caller
  End if
  Map correct device and block number from buffer
  parameters using partition table, dkunit, and
  dkbblock
  Map block number into cylinder group
  Call dksort to sort request into proper place in queue
  Call Dstart if the device is idle
End subroutine strategy
```

System Functions Dkunit, Dkbblock, and Dksort

These system functions are used to implement system standard device interleaving and request queue sorting.

The device interleaving is used to interleave a logical partition over two or more physical drives to balance I/O request loading on the drives. A secondary use of this function is to combine two drives into a larger, single logical partition.

The request queue sorting is used to provide a standard queue optimization. The standard algorithm used is up-elevator (C-Scan) with read preference. This algorithm gives better I/O throughput than FIFO, but allows significant unfairness to be introduced between multiple disk-bound processes. Although this is a good general algorithm, other algorithms produce fairer access to the disk and less erratic response times, while only reducing the total disk throughput by a small margin.

Subroutine Dstart

The purpose of this routine is to translate the parameters in the buffer header into command parameters for the disk controller. The I/O request is then started by the controller. When the request is done, the hardware generates an interrupt, which then calls the Dinterrupt routine. The interrupt routine processes the completion or retry of the last request and calls start to get the disk controller going again.

```
Begin subroutine start
  If request queue is empty, then
    Return
  End if
  Mark device as busy
  If buffer is raw_buffer and command is format, then
    Build format command
```

```
    Start format operation
    Return
  End if
  Build transfer command
  Start transfer operation
  Update metrics
End subroutine start
```

Subroutine Dinterrupt

The interrupt first checks for error on the transfer. If an error is found and there have not been too many retries, then the last request is restarted or marked as a hard error.

If the transfer was correct, then the buffer is marked done and start is called for the next request.

```
Begin subroutine interrupt
  If device was idle, then
    Log false interrupt if necessary
    Return
  End if
  If a device error, then
    If a soft error and only a few retries, then
      Clear error condition, if necessary
      Restart controller
      Return
    End if
    Flag error for request
  End if
  Mark request as done
  If more requests in queue, call start
End subroutine interrupt
```

The Simple Serial Terminal Driver

The basic form of a single line terminal driver is shown in Figure 3 (page 46). This driver is the minimal form for a single-character-at-a-time UART with interrupts handled in C code. In practice the Version 7 drivers are significantly more complex, particularly after performance optimization using software pseudo-dma. System 3 and System 5 drivers have additional complexity because device-specific output processing from the tty generic routines is transferred to the driver. It can be expected that even later versions will continue this trend to better support remote network terminals, multi-windowed terminals, and bit-mapped graphics terminals.

Subroutine Copen

```
Begin subroutine copen(minor_dev)
  If not a legal minor_device, then
    Post an error
    Return
  End if
  Initialize tty queuing/state structure
  If first open on device, then
    Set open flag
    Set default initial state flags
    Call system function ttychars for initialization
  End if
  Initialize hardware interface and interrupts
  Call system function ttyopen with queuing structure to
  complete open sequence
End subroutine
```


DIGITAL RESEARCH COMPUTERS

(214) 225-2309



STB Systems, Incorporated

IBM PC ADD ON BOARDS

NOW IN STOCK!

RIO PLUS II™ — Multi-function board with 2 RS232 serial ports (1 standard, 1 optional), parallel I/O port, game port and clock/calendar. Comes with 64K memory (expandable to 384K) and PC Accelerator.

STB-RIO + II 64 List \$395 **\$286.00**

GRAPHIX PLUS II™

Multi-function video board featuring monochrome text, full screen monochrome graphics, RGB color, composite b&w video, a parallel port, a light pen interface, and an upgradable clock option.

STB-GRAX List \$495 **\$363.00**

SUPER RIO™

RAM-I/O multi-function board with 64K memory (upgrad-able to 768K), two serial ports, parallel printer port, game port and clock/calendar.

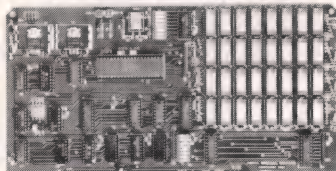
STB-SRIO-64 List \$419 **\$294.00**

All above boards come with "PC Accelerator" which is a RAM Disk Emulator and Printer Spooler that vastly increases your processing speed. All boards fully assembled and tested, and are warranted by the manufacturer for 1 year!

SPECIAL OFFER: Buy any two or more of the above boards and take an additional \$10 per board discount.

256K S-100 SOLID STATE DISK SIMULATOR!
WE CALL THIS BOARD THE "LIGHT-SPEED-100" BECAUSE IT OFFERS AN ASTOUNDING INCREASE IN YOUR COMPUTER'S PERFORMANCE WHEN COMPARED TO A MECHANICAL FLOPPY DISK DRIVE.

PRICE CUT!



BLANK PCB
(WITH CP/M* 2.2
PATCHES AND INSTALL
PROGRAM ON DISKETTE)
\$69.95

(8203-1 INTEL \$29.95)

- FEATURES:**
- * 256K on board, using + 5V 64K DRAMS.
 - * Uses new Intel 8203-1 LSI Memory Controller.
 - * Requires only 4 Dip Switch Selectable I/O Ports.
 - * Runs on 8080 or Z80 S100 machines.
 - * Up to 8 LS-100 boards can be run together for 2 Meg. of On Line Solid State Disk Storage.
 - * Provisions for Battery back-up.
 - * Software to mate the LS-100 to your CP/M* 2.2 DOS is supplied.
 - * The LS-100 provides an increase in speed of up to 7 to 10 times on Disk Intensive Software.
 - * Compare our price! You could pay up to 3 times as much for similar boards.

\$259.00

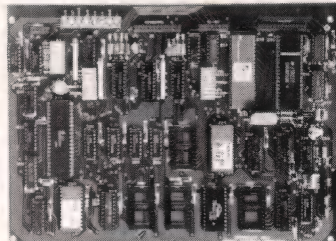
#LS-100 (FULL 256K KIT)

THE NEW ZRT-80

CRT TERMINAL BOARD!

A LOW COST Z-80 BASED SINGLE BOARD THAT ONLY NEEDS AN ASCII KEYBOARD, POWER SUPPLY, AND VIDEO MONITOR TO MAKE A COMPLETE CRT TERMINAL. USE AS A COMPUTER CONSOLE, OR WITH A MODEM FOR USE WITH ANY OF THE PHONE-LINE COMPUTER SERVICES.

- FEATURES:**
- * Uses a Z80A and 6845 CRT Controller for powerful video capabilities.
 - * RS232 at 16 BAUD Rates from 75 to 19,200.
 - * 24 x 80 standard format (60 Hz).
 - * Optional formats from 24 x 80 (50 Hz) to 64 lines x 96 characters (60 Hz).
 - * Higher density formats require up to 3 additional 2K x 8 6116 RAMs.
 - * Uses N.S. INS 8250 BAUD Rate Gen. and USART combo IC.
 - * 3 Terminal Emulation Modes which are Dip Switch selectable. These include the LSI-ADM3A, the Heath H-19, and the Beehive.
 - * Composite or Split Video.
 - * Any polarity of video or sync.
 - * Inverse Video Capability.
 - * Small Size: 6.5 x 9 inches.
 - * Upper & lower case with descenders.
 - * 7 x 9 Character Matrix.
 - * Requires Par. ASCII keyboard.



BLANK PCB WITH 2716
CHAR. ROM, 2732 MON. ROM

\$59.95

SOURCE DISKETTE - ADD \$10

SET OF 2 CRYSTALS - ADD \$7.50

WITH 8 IN.
SOURCE DISK!
(CP/M COMPATIBLE)

\$129.95

ZRT-80

(COMPLETE KIT,
2K VIDEO RAM)

64K S100 STATIC RAM

\$179.00
KIT

NEW!

LOW POWER!

150 NS ADD \$10

BLANK PC BOARD
WITH DOCUMENTATION
\$49.95

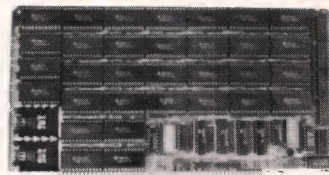
SUPPORT ICs + CAPS
\$17.50

FULL SOCKET SET
\$14.50

FULLY SUPPORTS THE
NEW IEEE 696 S100
STANDARD
(AS PROPOSED)

FOR 56K KIT \$165

ASSEMBLED AND
TESTED ADD \$50



FEATURES: PRICE CUT!

- * Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- * Fully supports IEEE 696 24 BIT Extended Addressing.
- * 64K draws only approximately 500 MA.
- * 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- * SUPPORTS PHANTOM (BOTH LOWER 32K AND ENTIRE BOARD).
- * 2716 EPROMs may be installed in any of top 48K.
- * Any of the top 8K (E000 H AND ABOVE) may be disabled to provide windows to eliminate any possible conflicts with your system monitor, disk controller, etc.
- * Perfect for small systems since BOTH RAM and EPROM may co-exist on the same board.
- * BOARD may be partially populated as 56K.

64K SS-50 STATIC RAM

\$149.95
(48K KIT)

NEW!

LOW POWER!

RAM OR EPROM!

BLANK PC BOARD
WITH
DOCUMENTATION
\$52

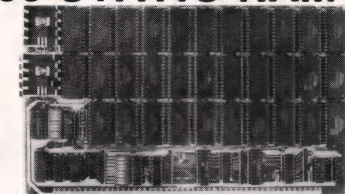
SUPPORT ICs + CAPS
\$18.00

FULL SOCKET SET
\$15.00

56K Kit \$169

64K Kit \$199

ASSEMBLED AND
TESTED ADD \$50



- FEATURES:**
- * Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
 - * Fully supports Extended Addressing.
 - * 64K draws only approximately 500 MA.
 - * 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
 - * Board is configured as 3-16K blocks and 8-2K blocks (within any 64K block) for maximum flexibility.
 - * 2716 EPROMs may be installed anywhere on Board.
 - * Top 16K may be disabled in 2K blocks to avoid any I/O conflicts.
 - * One Board supports both RAM and EPROM.
 - * RAM supports 2MHZ operation at no extra charge!
 - * Board may be partially populated in 16K increments.

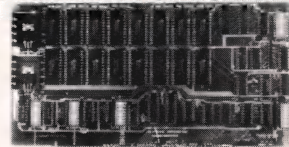
32K S100 EPROM/STATIC RAM

NEW!

FOUR FUNCTION BOARD!

NEW!

EPROM II
FULL
EPROM KIT
\$69.95
A&T EPROM
ADD \$35.00



BLANK
PC BOARD
WITH DATA
\$39.95

SUPPORT
IC'S
PLUS CAPS
\$16

FULL
SOCKET SET
\$15

We took our very popular 32K S100 EPROM Card and added additional logic to create a more versatile EPROM/RAM Board.

FEATURES:

- * This one board can be used in any one of four ways:
A. As a 32K 2716 EPROM Board
B. As a 32K 2732 EPROM Board (Using Every Other Socket)
C. As a mixed 32K 2716 EPROM/2K x 8 RAM Board
D. As a 32K Static RAM Board
- * Uses New 2K x 8 (TMM2016 or HM6116) RAM's
- * Fully Supports IEEE 696 Buss Standard (As Proposed)
- * Supports 24 Bit Extended Addressing
- * 200 NS (FAST!) RAM's are standard on the RAM Kit
- * Supports both Cromemco and North Star Bank Select
- * Supports Phantom
- * On Board wait State Generator
- * Every 2K Block may be disabled
- * Addressed as two separate 16K Blocks on any 64K Boundary
- * Perfect for MP/M* Systems
- * RAM Kit is very low power (300 MA typical)

PRICES
SLASHED!

32K STATIC RAM KIT — \$109.95

For RAM Kit A&T — Add \$40

Digital Research Computers

P.O. BOX 461565D • GARLAND, TEXAS 75046 • (214) 225-2309

TERMS: Add \$3.00 postage. We pay balance. Orders under \$15 add 75¢ handling. No. C.O.D. We accept Visa and MasterCard. Tex. Res. add 6-1/8% Tax. Foreign orders (except Canada) add 20% P & H. Orders over \$50, add 85¢ for insurance.

ALL SALES ARE MADE SUBJECT TO THE TERMS OF OUR 90 DAY LIMITED WARRANTY. A COPY OF THIS WARRANTY IS AVAILABLE FREE, ON REQUEST.

System Function Ttyopen

The ttyopen function does system-specific initialization that is common to all terminal drivers. This function generally includes establishing terminal ownership, process groups, and marking the queuing structure as open.

Subroutine Cclose

Begin subroutine cclose

Acquire queuing structure for this minor device

Call system function ttyclose with queuing structure
End subroutine

System Function Ttyclose

The ttyclose function performs the common close processing for terminal devices. This generally involves disassociating the device from any process group, waiting for characters in the output queue to finish transmitting, clearing the input queues, and marking the queue as closed.

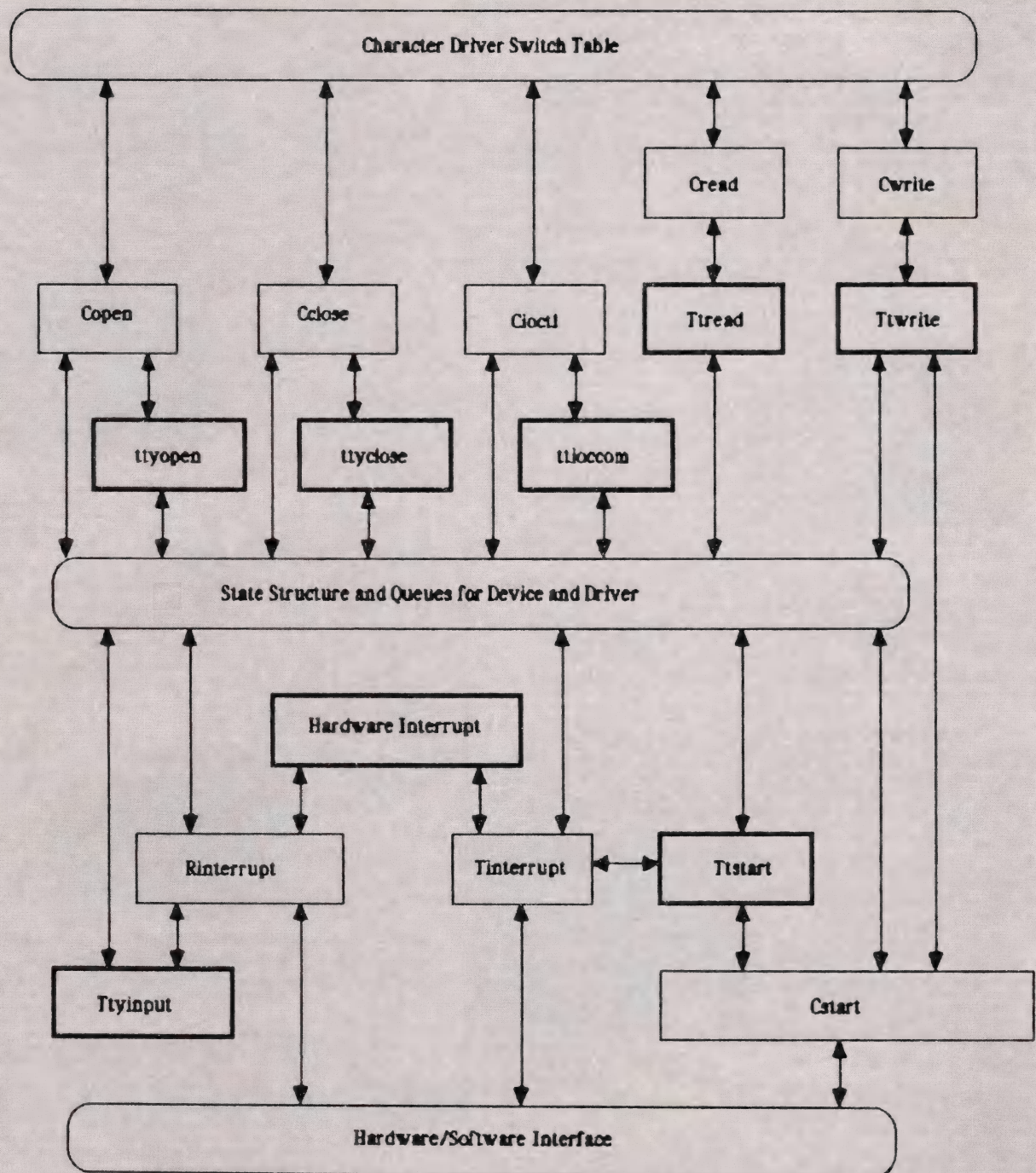


Figure 3
Simple Terminal Device Driver

Subroutine Cioctl

```
Begin subroutine cioctl(minor_device, cmd)
  Acquire queuing structure for this device
  Call system function ttioccom with queuing structure
    and cmd arguments
  If hardware parameters have changed for device, then
    Reprogram hardware to conform to new
      parameters
  End if
End subroutine
```

System Function Ttioccom

The ttioccom function performs the common processing to handle terminal ioctls reading and editing the hardware and software state variables for a serial port.

Subroutine Cread

```
Begin subroutine cread
  Acquire queuing structure for this minor device
  Call system function ttread with the queuing structure
End subroutine cread
```

System Function Ttread

The ttread function calls other routines to process the raw characters on the input queue. Any required editing is done, and the resulting input data is copied into the applications buffer. If more data is required and the input queue is empty, the routine sleeps while waiting for input.

Subroutine Cwrite

```
Begin subroutine cwrite
  Acquire queuing structure for this minor device
  Call system function ttwrite with queueing structure
End subroutine cwrite
```

System Function Ttwrite

The ttwrite function calls other routines to process the outgoing data. Terminal-specific handling is done for tabs, new lines, upper-case and lower-case characters, and delay processing. The routine sleeps as required on the output queue, copies data from the applications buffer to the output queue, and calls the start routine to keep the device busy.

Subroutine Cstart

```
Begin subroutine cstart
  If hardware is not ready, then
    Return
  End if
  If a character is on output queue, then
    Remove character from queue
    If the character is data, then
      Send the character out the serial line
    Else
      Do delay processing
    End if
  End if
End subroutine cstart
```

System Function Hardware Interrupt

By some hardware and/or software means the communica-

tions device presents an interrupt request to the system. Often special processing is required to save the hardware/software state at the time of the interrupt. After this state is saved, the proper driver interrupt processing routine is called.

Subroutine Tinterrupt

```
Begin subroutine tinterrupt
  Acquire queuing structure for device
  Call system function ttstart to do any special handling
    and then call driver's cstart routine
  If the output queue has enough free space, then
    Wake up any sleeping applications to fill it up
  End if
End subroutine
```

System Function Ttstart

The ttstart function does common processing before calling the device's start routine. The function generally includes checks to prevent starting a line that is stopped for some reason or is already busy.

Subroutine Rinterrupt

```
Begin subroutine rinterrupt
  Acquire queuing structure for device
  If there is data in receiver, then
    Get character from serial receiver
    Call system function ttyinput to process character
  End if
  Reset interrupt enable as required
End subroutine
```

System Function Ttyinput

The ttyinput function handles common processing of input characters at interrupt time. This generally includes queuing character echo, handling xon/xoff flow control, signals, end of file, and converting input case. The input is queued for edit processing by ttread. A wakeup of ttread is done as required—either on a special condition or on every character if in raw or cbreak mode.

Background

There are many pitfalls for device driver writers. To explore them in reasonable detail would require a three- to four-day seminar. Two interesting problems are common "mis"-features in many Unix systems currently being shipped.

Better Dsort Algorithms

To evaluate various candidate algorithms, readers must first abandon prior knowledge of what is best and then take a good look at the Unix environment and applications.

(1) A large number of studies have been done on disk queuing strategies for batch environments to optimize throughput. Response time and fairness are not addressed as key variables in those studies. Unfortunately, the optimal solutions for throughput have the worst response time and fairness qualities. Computer scientists as a whole tend not to reevaluate old trade-offs in the face of new requirements. The educational community continues to close the minds of newer computer scientists by teaching "truths" that are out of perspective with today's problems.

(2) Queuing algorithms that change the order of writes going to the disk prevent applications from using fail-safe updates to large data bases. With an unknown write ordering, it is impossible to back out certain update transactions after a system crash. FIFO handling of writes is a must.

(3) The standard Unix *dsort* gives preference to reads and defers all writes to improve response times. This is generally the best policy to prevent blocking on buffered output data. Unfortunately, the standard implementation also defers writes for swapping and paging as well. This ties up the swap and paging buffer headers for long times when several active readers can lock down the request queue. This results in step reductions or nonlinear decay in both system throughput and system efficiency as the disk queue length increases.

(4) Any algorithm that allows one or more processes to dominate the queue is bound to cause a wide variance in response times. Since slow response times are subjective (that is, the user remembers only the worst), then algorithms that generate wide variances are likely to create the "slow response time" conditions that the user will see and remember at random times. If the variance gets worse as the load increases, then the load will become more visible to the user than would normally be expected.

With these requirements in mind it is clear that the standard *dsort* is nonoptimal for a timesharing system. The standard up-elevator algorithm with deferred writes allows a single active reader of a sequentially allocated file to lock up the disk queue with readahead. This will continue for a maximum scheduling quantum of several seconds. Several such users can create service time delays to other processes in the disk queue of several times the scheduling quantum! If the system is swapping or paging at the same time, then swap and page requests are likely to be serviced once per few seconds—this allows the incore process hogging the queue to do so for even longer periods of time.

The typical up-down elevator algorithm is even worse. The requests for I/O at the ends of the service area get serviced only once per pass, while the middle gets serviced twice per pass. The result is a normal distribution for the probability of service and a process throughput as a function of the location of the file. Processes accessing data at the ends of the request area typically get between 5 – 50% of the normal throughput depending on queue length and access patterns. Note that for most Unix systems the swap/paging areas and the inodes/directories tend to be at one end of the service pattern. The typical result is *exec/fork* operations that are significantly degraded as the queue length increases, resulting in abnormal response times for simple commands. When swap/paging traffic is affected by this, the system suffers a large step reduction in efficiency and throughput as the swap/paging time exceeds the scheduling quantum.

To fix this behavior requires that lockdown be bounded to short bursts (some lockdown is desired to obtain higher throughputs), and that swapping or paging always get premium service.

The proper algorithm is one that sorts the disk queue in the reverse direction from that in which file data is sequentially allocated. This is down-elevator on most Unix systems. Second, writes on minor devices with file systems are queued FIFO and serviced after all outstanding read requests and

swap/page requests have been serviced.

A better variation of this is to service small batches of writes that fall into the natural down-elevator algorithm. Thus, on every pass a few writes are taken off the queue and released. This helps prevent the write-deferred buffer from clogging up the buffer pool. The writes are still done FIFO but do not cause a burst of nonoptimal FIFO traffic in some random access pattern.

Better Serial Interrupt Handling

Character-at-a-time, interrupt-driven servicing can use up large amounts of high-priority, nonschedulable cpu time. For most systems, the time to service an input keystroke in raw mode is between 800 and 3500 microseconds. This includes the time required to process that hardware interrupt, save system state, call the high-level receiver interrupt routine, echo the character, start the output, wake up the sleeping application, and do a context switch to run the process. Not included is the additional 2500 to 4000 microseconds that it takes to perform the read system call and schedulable kernel-level handling of the input data. At 2400 – 4800 baud, most systems are totally cpu bound and will lose incoming data. Normally humans don't type that fast, but repeat functions and multicharacter function keys do!

A common solution to this problem is to handle the receiver and transmit interrupts in a short assembly language interrupt routine. This routine uses a 50 – 200-character dedicated output buffer and a shared 300 – 2000-character input buffer. The tightly coded output interrupt routine requires 30 – 90 microseconds per character, and the normal output routines require 10 – 200 microseconds per character to keep the output buffer full. Similar numbers are true for input. Wakeups are done only on buffer full or empty. A watchdog timeout routine calls the input processor every few clock ticks to process incoming data when the buffer doesn't get full. The net effect is a 70 – 95% reduction in cpu time for serial communications traffic and significantly reduced chances of dropping input characters.

DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 193.

Dr. Dobbs says, "WE HAVE GOOD NEWS..."

"WINDOWS FOR C can offer you a convenient and reliable means of implementing windows in your text-handling programs..."

"The video output is fast and clean, showing no snow with either the graphics or monochrome display.

"WINDOWS FOR C is documented in well-written and readable English, and it appears to be all there.

"WINDOWS FOR C... is a very nice software package that has obviously been well thought out and very cleanly executed.

"WINDOWS FOR C is a professionally-oriented set of programming tools... that we can recommend."

— Ian Ashdown (*Dr. Dobbs' Journal*, November 1984)

Full support for:

- IBM PC/XT/AT plus compatibles
- All memory models of Lattice C, C1-C86, Mark Wm. C, Aztec C, Microsoft C, Desmet C (PC/MSDOS)

Plus:

- NEW: Driver interfaces for non-IBM displays
- Full source available

More than a window display system, WINDOWS FOR C is a video toolkit for all screen management tasks.

C SOURCE is provided for pop-up menus, viewing of multiple ASCII files within multiple windows, cursor control for vertical and horizontal scrolling, label printing, and text-mode bar graphs.

An object-code library of over 50 fully documented building-block subroutines allows you to construct functions to suit your needs.

Once you've used WINDOWS FOR C, you will wonder how you managed without it.

With WINDOWS FOR C you can:

- Establish any number of windows; clear, write, and control attributes of each window independently; write with word wrap and auto scroll; control all IBM color capabilities.
- Overlay and then restore screens, highlight selected text, format and print with windows, and save windows to memory or disk. Memory management is automatic.

Don't wait. Order now.



The Good News...

WINDOWS FOR C™

ADVANCED SCREEN MANAGEMENT

WINDOWS FOR C
(specify compiler & version)

\$195

Demo disk and manual
(applies toward purchase)

\$ 30

Dealer inquiries welcome

A PROFESSIONAL SOFTWARE TOOL FROM

CREATIVE SOLUTIONS

21 Elm Ave., Box D12 • Richford, VT 05476

802-848-7738

Master Card & Visa Accepted
Shipping \$2.50
VT residents add 4% tax

Circle no. 27 on reader service card.

A Unix Internals Bibliography

by John Rogers

"You are in a dark, musty, source code directory, with many mysteriously named source files around you . . ."

A number of books and manuals describe how to use the Unix operating system. However, Unix internals documentation is pretty scattered (where it exists at all). This is unfortunate, especially for those of us who work with Unix—the lack of design documents makes our jobs harder. Some public domain Unix-like operating systems (GNU and XINU) are in the works for the rest of the world, and a number of worthwhile ideas in Unix are adaptable to other systems. In the next few years, documentation on the Unix algorithms and data structures will become more valuable to many people than the actual source code licenses for Unix.

how something works is to "grep for it" in the source code (in fact, there are t-shirts that say "grep for it!"). This isn't always desirable, however, because some of us don't have access to the source code, or, if we do, the source isn't particularly lucid. A comment in the code even says, "You aren't supposed to understand this"!

I've since given up on finding one comprehensive document, but I've become aware that bits and pieces of design documentation are available. Because most of this documentation is in fairly obscure places, I've put together a bibliography to help other people find the information they need.

Using this Bibliography

I have divided the main body of this article into seven subject areas: the kernel, the shell, compilers and language

You are supposed to understand this.

Most Unix gurus will tell you that, because Unix is small, written in a high-level language, and available with the source code, documentation isn't as necessary as it is for some other systems. However, Unix has enough code to make earning the title "Unix guru" quite a task. For the first year or two that I was working with Unix, I kept thinking "there has to be some sort of design document that Bell Labs hasn't released," but I haven't found it yet. The normal means of discovering

development tools, communications, the C runtime library, miscellaneous utilities, and regular expressions. This is followed by a list of references in alphabetical order by author. Although the organization may seem a little arbitrary, I don't think people will have problems finding what they want.

Within the main body of this article, books, papers, research reports, magazine articles, and the like are cited with the author's name in square brackets; complete bibliographical information appears in the list of references. In the few places where man page references appear, I use the standard form (e.g., LS(1) refers to the man page for "ls" in section 1 of volume 1 of the manual). I have adopted the numbering scheme used in almost every major release except System V (i.e., section 4 has device

John Rogers, 1740 California Street #6, Mountain View, CA 94041.

Copyright (c) 1984 by John Rogers (jr@foros1.UUCP). All rights reserved. Unix is a trademark of AT&T Bell Laboratories.

drivers, and section 5 has file formats); most other things are unchanged.

K&R, of course, refers to *The C Programming Language* by Kernighan and Ritchie (see [Kernighan and Ritchie78a] in the list of references). UPM is short for the *Unix Programmer's Manual*, which almost always comes in two volumes: the first contains the man pages, and the second contains a number of papers about Unix (some of which are referenced later). BSTJ is the *Bell System Technical Journal* (now the *AT&T Bell Laboratories Technical Journal*), available from Room 1J319, 101 J. F. Kennedy Parkway, Short Hills, NJ 07078. CSTR stands for *Computer Science Technical Report*, available (for free) from Room 2C-213, AT&T Bell Laboratories, Murray Hill, NJ 07974. CSRG is the Computer Science Research Group at U. C.—Berkeley.

The Kernel

A group of people at Purdue University wrote a public domain operating system named XINU (a recursive acronym for “XINU Is Not UNIX”); the source code has been published in [Comer]. Although XINU has a lot of the flavor of Unix (especially since it's written in C), it is not a complete time-sharing system by any means; there are no shells or utilities, for instance. For someone unfamiliar with Unix internals, this is a good place to get acquainted with the concepts used in the Unix kernel. For someone who has already read some of the source code to Unix's kernel, however, the XINU code is kind of disappointing.

A document [Lions77] put together for an operating systems course at the University of New South Wales in Australia describes the Unix kernel in some detail. Because it takes the form of annotations to the source code (by line number), it isn't much use without a source listing (see, for instance, [Lions76]). Unfortunately, the document is based on Version 6; there have been four major releases of Unix since then, so it's not exactly current.

[Lions77] presents another problem for people without source licenses. Because it contains a lot of details about the kernel, you're supposed to have a Unix source license to get a copy. Nev-

ertheless, lots of bootleg copies are floating around. For that reason and for the sake of completeness, I'm mentioning this document here. For more information on the whole topic, see [Lions78], which is available to the public.

Allocation of Directory Entries

Allocation of directory entries in most releases of Unix is pretty trivial; 4.2 BSD is the only major exception. See the DIR(5) man page for details on that.

Allocation of Disk Blocks and Inodes

[McKusick, Joy, Leffler, and Fabry] discusses allocation of data blocks and inodes in 4.2 BSD. In other releases of Unix, this allocation tends to be simpler; see the FILSYS(5), FS(5), or FS(4) man page entry for details.

Bootting

[Christian] has a pretty good piece (section 19.5) on bootting, titled “Bootting, Process 0, Process 1.” Since bootting seems to be different in every release and on every machine, read this with a grain of salt. Section 8 of your Unix manual probably has more detail and accuracy than anything else.

Device Drivers: General

If you have no idea how Unix's device drivers fit in with the rest of the kernel, you might want to read [Comer]; it gives you the flavor of the subject, although a fair number of the details are different from Unix (remember, he's writing about XINU).

Once you have a rough idea of how device drivers fit into Unix (i.e., via major and minor device numbers and the device “switches”), then [Ritchie78] is the thing to read: it discusses the device-switch tables as well as the kernel routines nulldev(), cpass(), passc(), iomove(),getc() and putc() (not to be confused with the standard I/O routines of the same names), sleep(), wakeup(), timeout(), spl3() and so on, bread(), getblk(), breada(), brelse(), bwrite(), bawrite(), bdwrite(), geterror(), and physio().

Because [Ritchie78] is terse (like most of Unix), you might want to read something that goes a little slower and gives examples. [McNamara, Vaish,

and Bryant], which examines various aspects of device drivers in detail and gives two pages of pseudo-code for a block device driver, discusses sleep(), wakeup(), timeout(), physio(), spl0-7(), disksort(), iowait(), and deverror().

If you're actually going to write a device driver, then you may need all the help you can get. [Nystrom] is pretty much the complete authority, with hundreds of pages, lots of source code, and reprints of [Hickman83a] and [Hickman83b]. Unfortunately, you also need a Unix source license, and I'm not sure if that is available other than by taking a seminar from International Technical Seminars (see the list of references).

One last suggestion: According to an article on Usenet by John Levine at INTERACTIVE Systems (john@ima.UUCP), “the PC/IX manual includes a fairly informative section on writing device drivers.” I haven't seen a copy yet.

Device Drivers: Block

[Ritchie78] discusses buffer headers and associated routines. [Hickman83a] is another source of information. As mentioned earlier, [McNamara, Vaish, and Bryant] gives two pages of pseudo-code for a block device driver.

Device Drivers: Character

[Thompson] and [Ritchie78] look at “character lists” (really queues) and the routines that access them.

Directories

Chapter 8 in [Kernighan and Ritchie78a] discusses the format of directories. In [Kernighan and Pike] (page 59) is the comment that an inode number of zero indicates an empty directory entry.

exec()

[Johnson and Ritchie77] mentions that exec() builds the argument list for a new program by allocating memory on the stack.

File I/O

[Thompson] in section 4.1 gives some information on the processing done by open(), including descriptions of the per-user file table, the system file table,

and the inode table; [Christian] in section 19.6 more or less repeats the same information. [Ritchie78] discusses opens on devices. See also `namei()` below—`namei()` is called indirectly when the user does an `open()` or `creat()`.

getty

[Christian] in section 19.5 examines the relationships between `init`, `getty`, and `login`, though not in much detail. Note that the parameters passed to `getty` tend to vary from one implementation of Unix to the next. See the `GETTY(8)` or `GETTY(1M)` man page for details.

iget() and iput()

[Wales] discusses `iget()` and `iput()`. `iget()` is called to locate or read in an in-core copy of an inode, given its inode number; reference counts are used to keep track of how many processes are using a given inode. `iput()` decrements the reference count and, when the reference count goes to zero, frees the in-core copy, writing it out to disk if it has been modified.

[Wales] also discusses the hashing used to find in-core inodes—and the use of a singly linked list if collisions occur. [Lankford] mentions that, as of System V and 4.1 BSD, inode table entries are hashed (in releases prior to those, the table was searched sequentially), and [Ritchie79] notes that in the original PDP-7 Unix, `iget()` left the inode it found in a constant location.

init

Section 6.6 in [Ritchie and Thompson] gives a superficial overview of how `init` works (ignoring both `getty` and `login`). The `INIT(8)` (or `INIT(1M)`) man page provides more detail. Of course, `init` keeps changing from one release to the next. [Gauthier] in section 19.5 supplies a great deal of detail about how process 1 is created “from scratch.”

Memory Management

[Johnson and Ritchie 77] includes a short discussion of various aspects of memory management in the Unix kernel. [Thompson] observes that “the swapping process is the only process that waits for primary memory to become available.”

mount()

[Ritchie and Thompson] in section IV and [Thompson] in section 4.2 discuss how the mount table is built and how `namei()` uses it.

namei()

This routine in the kernel converts from a path name to an inode number. It is mentioned in [Leffler, Karels, and McKusick], along with the fact that 4.2 BSD has `namei()` caching—with a table of names, inode numbers, device numbers, and pointers to inode table entries. This same source explores the logic used by `namei()`. Sections 4.1 and 4.2 in [Thompson] discuss various aspects of the translation of the name to the inode number. (See also `iget()` and `iput()` above.)

Process Table

[Leffler, Karels, and McKusick] mentions that the `newproc()` routine in the kernel does a linear search of the process table to allocate an ID for each new process. [Goodwin] describes in significant detail various algorithms for allocating new process IDs.

[Leffler, Karels, and McKusick] also gives a list of other kernel routines and/or system calls that do sequential searches of the process table, as well as the reasons for each search. The routines are: `exit()`, `wait()`, `fork()`, `newproc()`, `kill()`, `gsignal()`, `schedcpu()`, and `sched()`.

Read-ahead

The Unix kernel implements a read-ahead scheme where the kernel tries to predict which disk blocks will need to be read in and to read them in before they are actually requested by a user program. This is discussed in [Ritchie78]. [Goodwin] proposes an improvement on the algorithm.

Scheduling

[Gauthier] in section 19.3 looks briefly at scheduling, as does [Thompson] in section 2.3. You can glean other details from a careful reading of the `PS(1)` man page.

u page

Much of the data about a process is stored in the u page (so called because the global variable with its address is named “u”); this is discussed in [Ritchie78] and [Goodwin], where it is

called the per process data area, and in [Thompson], where it is called the system data segment. Note that a u page is different from a process table entry (of which there is also one per process): the u page is swapped to disk, while the process table entries are always resident in main memory.

Update Process

[Gauthier] on page 204 mentions the update process (with `/etc/update` showing in the “ps” example). On page 213 is the comment: “the `/etc/update` program forces disk updates every thirty seconds and should be run at every installation.”

The Shell

[Ritchie and Thompson] in section 6.5 describes the shell’s use of `exec()`, `fork()`, `wait()`, and `read()`; how it handles pipes and I/O redirection; and how shell scripts work.

Background Processes

[Joy] shows how the C shell handles its table of background processes.

cd

Most man pages for `CD(1)` (or `CHDIR(1)`) mention that the command must be built into the shell; [Ritchie79] gives an interesting historical note on how the authors of Unix discovered this.

Filename Expansion

See the section on regular expressions (page 56).

I/O Redirection

[Holt] on page 169 gives a sketchy description of how this is handled.

Running Programs

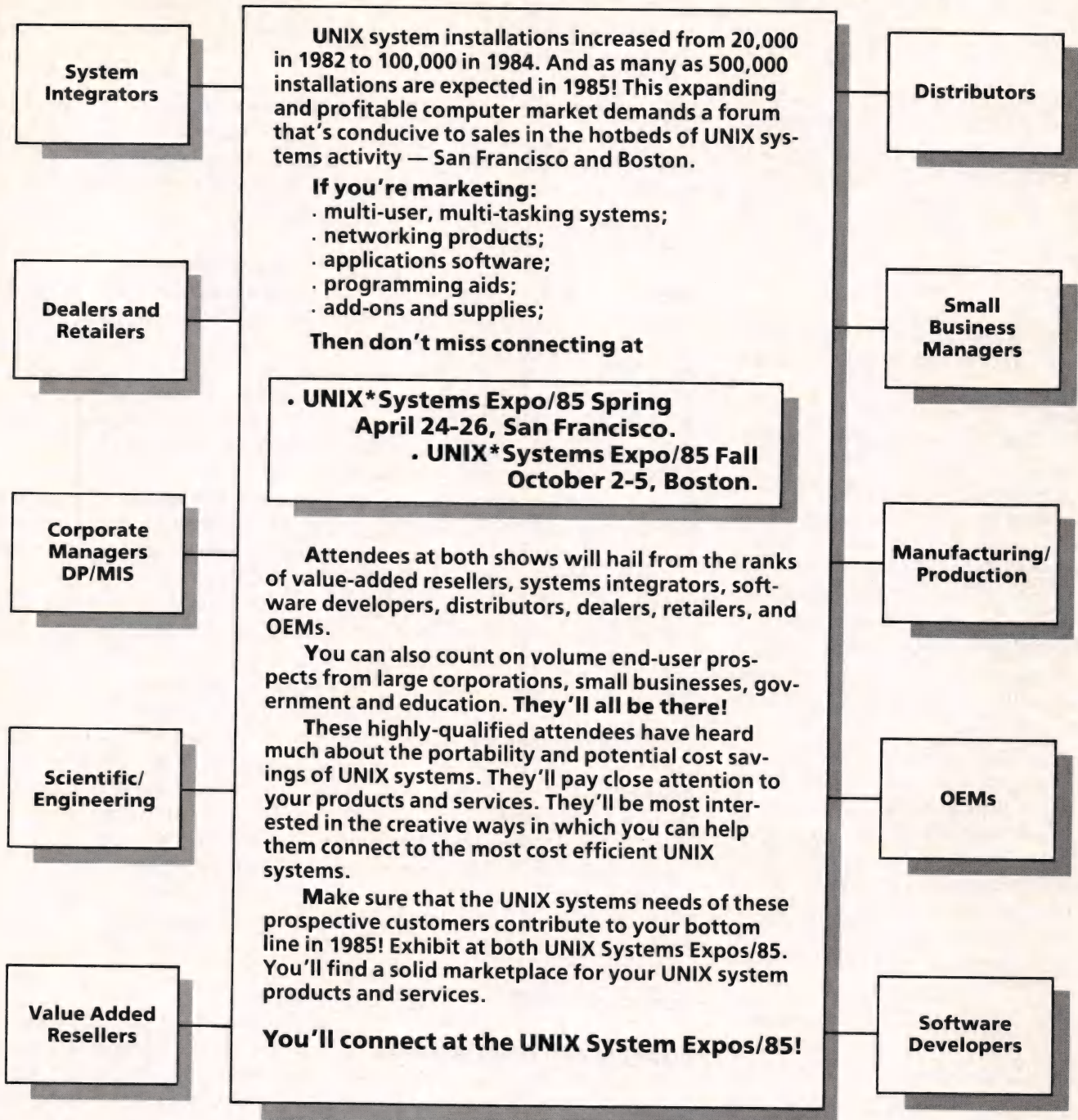
[Holt] on pages 167-168 presents some pseudo-code for the steps a shell goes through when it creates a subshell and runs a given program; it does not, however, describe how shell scripts are handled when they are run. See also the discussion of the `system()` subroutine in the section on the C runtime library (page 54.)

Compilers and Language Development Tools

c++

[Johnson and Lesk] mentions that the

How To Make Profitable UNIX* Systems Connections



Exclusive productions of
Computer Faire, Inc./
A Prentice-Hall Company.
181 Wells Avenue
Newton, MA 02159
617/965-8350
611 Veterans Boulevard
Redwood City, CA 94063
415/364-4294

*UNIX is a trademark of AT&T
Bell Laboratories.

Yes, I'm interested in
hearing more about
UNIX* Systems Expo/85

- ☐ San Francisco
☐ Boston
☐ Both

Name _____

Title _____

Company _____

Address _____

City/State/Zip _____

Phone _____

Computer Faire, Inc. 181 Wells Avenue Newton, MA 02159

C preprocessor is written using yacc and lex. Martin Minow posted a complete public domain cpp to Usenet this summer, and another partial implementation ("p") is given with full source code listings in [Schreiner].

lex

[Johnson and Lesk] states that lex is written using yacc. [Aho and Ullman] has quite a bit of the theory concerning lex in its chapter 3. See also the section on regular expressions. A public domain version of lex is available from Decus.

lint

[Johnson78a] touches on the design of lint in the section on implementation.

pcc

[Johnson and Lesk] mentions that pcc is written using yacc and lex.

[Johnson79] is a long, detailed introduction to the guts of the portable C compiler. It discusses parsing, symbol table handling, expression trees, and register allocation. It gives a couple of examples of the templates used to rewrite expression trees and eventually to generate code, as well as a number of the function and source filenames for the compiler.

[Leffler] is an exhaustive document (100 pages, although some of it applies only to the Harris /6). It examines the organization of the compiler in great detail, giving source file and function names throughout. Surprisingly, it gives very few examples of expression trees and rewriting templates. It is very good nonetheless.

The Ritchie (PDP-11) C Compiler

[Ritchie76] is, of course, the authoritative paper on the PDP-11 C compiler. (Most C compilers are derived from pcc, but there are a few. . .)

[Pammett] discusses porting the Ritchie C compiler to the TI990. For a paper of this size (200+ pages), it says amazingly little about how the compiler works. An appendix, however, gives a detailed description of the code tables.

yacc

[Johnson78b] examines how the parser built by yacc operates and provides a detailed example of a parse ta-

ble. The theory behind yacc gets attention in [Aho and Ullman], which contains pointers to other papers and books on the subject as well. This book also presents the grammar and parse table for a subset of eqn.

A public domain implementation of yacc is available from Decus.

Communications

Networking in 4.2 BSD

[Leffler, Joy, and Fabry] devotes 29 pages to the nitty-gritty of sockets and protocol handling in the 4.2 kernel.

UUCP

[Nowitz] discusses queuing, work file formats, control files, and a bunch of the "setup" protocol, but doesn't offer much about the main file transfer protocol (the "g" protocol). [Nowitz and Lesk] looks at the initial handshaking.

For those who are really interested in how the "g" protocol fits into UUCP, Piet Beertema of CWI in Amsterdam (piet@mccvax.UUCP) wrote another protocol—the "f" protocol for use on X.25 networks—that "drops into" UUCP. The source code for this protocol is public domain and was posted to Usenet this summer.

The C Runtime Library

bsearch()

bsearch(), the binary search routine from System V, uses algorithm B in section 6.2.1 of [Knuth] (according to the man page).

Calling Sequences

[Johnson and Ritchie 81] discusses these in great detail.

ctype (isalpha(), isascii(), etc.)

This is pretty trivial, but a paper was actually published that talks about how the CTYPE(3) macros work; see [Gimpel] if you're interested.

dbm(3)

The algorithm used by the data base routines dbmopen(), etc., which are documented in the DBM(3) man page, is described in [Fagin, et al.], [Carter and Wegman], and possibly in the November 1982 *BSTJ* (part two).

hsearch(), hcreate(), hdestroy()

These hash table routines (in System V) use algorithm D from section 6.4 of [Knuth] (according to the man page).

isatty()

[Arnold] mentions that isatty() simply calls gtty() and checks the return value.

lsearch(), lfind()

These linear search routines (lfind() was added in System V, release 2) use algorithms from section 6.1 of [Knuth].

Memory Allocation

[Kernighan and Ritchie 78a] gives listings of alloc() and free() in section 8.7. Various algorithms are used in different releases of Unix: System V has two different versions of malloc(), and 4.2 BSD supposedly has a power-of-two allocation scheme.

printf(), fprintf(), and sprintf()

For source listings of more or less complete printf() functions, see [Hendrix] or [Comer].

Some versions of the printf() family call an internal routine named _doprnt(): see, for instance, [Kernighan and Pike], page 189. Eric Kiebler (eric@washu.UUCP) posted this to Usenet a while back:

"BEWARE the _doprnt
code, my son!
The bits that twitch;
the types that clash!
Use the portable
varargs stuff,
Avoid the _doprnt's
teeth that gnash!"

qsort()

This uses the quicker-sort algorithm, which has been described everywhere including section 5.2.2 of [Knuth]; see especially page 123 where qsort() chooses a trial median from the center of the list.

stdio

A great deal of detail on the workings of the standard I/O library is given in chapter 8 of [Kernighan and Ritchie 78a].

Strings (strcmp(), strtok(), etc.)

A public domain implementation of the various string functions (literally

#68 Volume VII, No. 6:

Multi-68000 Personal Computer—PDP-1802, Part One—Improved LET for LLL Basic—CP/M Print Utility.

#69 Volume VII, No. 7:

IBM-PC Issue: CP/M-86 vs. MSDOS (A Technical Comparison)—Hi-Res Graphics on the IBM-PC—PDP-1802, Part II—Review of Word Processors for IBM.

#70 Volume VII, No. 8:

Argum "C" Command Line Processor—SEND/RECEIVE File Transfer Utilities—Intel's 8087 Performance Evaluation.

#71 Volume VII, No. 9:

FORTH Issue: Floating-Point Package—H-19 Screen Editor—Relocating, Linking Loader—Z8000 Forth—Forth Programming Style—8086 ASCII-Binary Conversion Routines—CP/M Conditional SUBMIT.

#72 Volume VII, No. 10:

Portable Pidgin for Z80—68000 Cross Assembler, Part I—MODEM and RCP/MS—Simplified 68000 Mnemonics—Nested Submits—8086/88 Trig Lookup.

#73 Volume VII, No. 11:

Wildcard UNIX Filenames—Tests for Pidgin—68000 Cross Assembler Listing, Part 2—Adding More BDOS Calls—The Perfect Hash—BASIC Memory Management—Benchmarks for CP/M-86 vs. MSDOS, and the 8087.

#76 Volume VIII, Issue 2:

PISTOL, A Forth-like Portably Implemented Stack Oriented Language—Program Linkage by Coroutines. Forth to BASIC—Linking CP/M Functions to Your High-Level Program—Concurrent CP/M-86—CP/M-80 Expansion Card for the Victor 9000—REVAS Disassembler.

#77 Volume VIII, Issue 3:

The Augusta P-Code Interpreter—A Small-C Operating System—6809 Threaded Code: Parametrization and Transfer of Control—A Common-Sense Guide to Faster, Small BASIC—A Fundamental Mistake in Compiler Design—Basic Disk I/O, Part I.

#78 Volume VIII, Issue 4:

RECLAIM Destroyed Directories—Binary Magic Numbers—8080 Fig-Forth Directory & File System—"SAY" Forth Votrax Driver—TRS-80 8080 to Z80 Translator—Basic Disk I/O, Part II.

#79 Volume VIII, No. 5:

The Augusta Compiler—A Fast Circle Routine—Enhancing the C Screen Editor—Shifts and Rotations on the Z80—The SCB, TSX, and TXS Instructions of the 6502 and 6800—MS-DOS vs. CP/M-86—Controlling MBASIC—The Buffered Keyboard—IBM PC Character Set Linker—Flip Utility for the IBM PC.

#80 Volume VIII, Issue 6:

Fast Visibility Algorithms—B-Tree ISAM Concepts—CP/M BDOS AND BIOS Calls for C—Serial Expansion in Forth—Fast Matrix Operations in Forth, Part I—Yes, You Can Trace Through BDOS—Julian Dates for Microcomputers—8088 Addressing Modes—8088 Line Generator—CP/M Plus.

#81 Volume VIII, Issue 7:

The Augusta Compiler, continued—RED: A Better Screen Editor, Part I—Anatomy of a Digital Vector and Curve Generator—Fast Matrix Operations in Forth, Part II—The AGGHHH Program—MBOOT Revisited—CP/M Plus Feedback—MS-DOS Rebuttal—68000 Tools—Sizing Memory on the IBM PC.

#82 Volume VIII, Issue 8:

Serial-to Parallel: A Flexible Utility Box—McWORDER: A Tiny Text Editor—And Still More Fifth Generation Computers—Specialist Symbols and I/O Benchmarks for CP/M Plus—CP/M Plus Memory Management—Zero Length File Test—PAUSEIF, QUITIF, and now SKIPIF—ACTxx Cross Assemblers.

#83 Volume VIII, No. 9:

FORTH ISSUE: Forth and the Motorola 68000—Nondeterministic Control Words in Forth—A 68000 Forth Assembler—GO in Forth—Precompiled Forth Modules—Signed Integer Division—Some Forth Coding Standards—The Forth Sort.

#84 Volume VIII, No. 10:

Unix to CP/M Floppy Disk File Conversion—A Small-C Help Facility—Attaching a Winchester Hard Disk to the S-100 Bus—Using Epson Bit-Plot Graphics—8086/88 Function Macros—Auto Disk Format Selection—CP/M Plus Device Tables.

#85 Volume VIII, Issue 11:

A Kernel for the MC68000—A DML Parser—Towards a More Writable Forth Syntax—Simple Graphics for Printer—Floating-Point Benchmarks.

#86 Volume VIII, Issue 12:

Faster Circles for Apples—Cursor Control for Dumb Terminals—Dysan's Digital Diagnostic Diskette—Interfacing a Hard Disk Within a CP/M Environment—The New MS-DOS EXEC Function.

#87 Volume IX, Issue 1:

A Structured Preprocessor for MBASIC—A Simple Window Package—Forth to PC-DOS Interface—Sorted Diskette Directory Listing for the IBM PC—Emulate WordStar on TOPS-20—More on optimizing compilers—The PIP mystery device contest.

#88 Volume IX, Issue 2:

Telecommunications Issue: Micro to Mainframe Connection—Communications Protocols—Unix to Unix Network Utilities—VPC: A Virtual Personal Computer for Networks—PABX and the Personal Computer—BASIC Language Telecommunications Programming—U.S. Robotics S-100 Card Modem.

#89 Volume IX, Issue 3:

RSA: A Public Key Cryptography System, Part I—Introduction to PLUC: Programming Language for Compilers—Program Design Using Pseudocode—More on Binary Magic Numbers—How fast is CP/M Plus?—CP/M 2.2 BIOS Function: SELDSK—The results of the Floating-Point benchmark.

#90 Volume IX, Issue 4:

Optimizing Strings in C—Expert Systems and the Weather—RSA: A Public Key Cryptography System, Part II—Several items on CP/M Plus, CP/M v2.2 Compatibility—BDOS Function 10: Vastly Improved—More on MS-DOS EXEC Function—Low-Level Input-Output in C.

#91 Volume IX, Issue 5:

Introduction to Modula-2 for Pascal Programmers—Converting Fig-Forth to Forth-83—Sixth Generation Computers—A New Library for Small-C—Solutions to Quirks in dBASE II.

#92 Volume IX, Issue 6:

CP/M on the Commodore 64—dBASE II Programming Techniques—First Chinese Forth: A Double-Headed Approach—cc-A Driver for a Small-C Programming System—A New Library for Small-C (Part II)—Comments on Sixth Generation Computers—Review of Turbo Pascal.

#93 Volume IX, Issue 7:

RSX under CP/M Plus—p-A Small-C Preprocessor—A Simple Minimax Algorithm—Languages and Parentheses (A Suggestion for Forth-like Languages)—Comments on assembly language development packages, RSX to patch CP/M 2.2 with CP/M, iRMS-86 for the IBM PC, C Programming Tools.

#94 Volume IX, Issue 8:

SCISTAR: Greek and Math Symbols with WordStar—A File Comparator for CP/M Plus—Designing a File Encryption System—A Small-C Concordance Generator.

#95 Volume IX, Issue 9:

Forth Special Issue!—File Maintenance in Forth—Forth and the Fast Fourier Transform—Computing with Streams—A Forth Native-Code Cross Compiler for the MC68000—The FVG Standard Floating-Point Extension—CP/M Plus: Interbank Memory Moves Without DMA—ways to make C more powerful and flexible.

#96 Volume IX, Issue 10:

More dBASE II Programming Techniques—Simple Calculations with Complex Numbers—GREG: C: A Unix-like Generalized Regular Expression Parser—An optimization scheme for compilers, MSDOS 2.0 Filters, Sizing RAM under MSDOS, Two programming systems illustrating Runge-Kutta integration.

#97 Volume IX, Issue 11:

Adding Primitive I/O Functions to mULISP—Program Monitor Package: Using Interrupts to Instrument Applications—CP/M 2.2 Goes PUBLIC—A Guide to Resources for the C Programmer—RESORT.

TO ORDER:

Send \$3.50 per issue to: **Dr. Dobb's Journal**, 2464 Embarcadero Way, Palo, Alto, CA 94303

Please send me the issue(s) circled: **68 69 70 71 72 73**

76 77 78 79 80 81 82 83 84 85 86

87 88 89 90 91 92 93 94 95 96 97

I enclose \$_____ (U.S. check or money order).

Outside the U.S., add \$.50 per issue.

I have read the postal instructions and understand that I will not receive my order unless I have sent the correct payment amount.

Please charge my: ☐ Visa ☐ M/C ☐ Amer. Exp.

Card No. _____ Exp. Date _____

Signature _____

Name _____

Address _____

City _____

State _____ Zip _____

Availability on first come/first serve basis. Outside the U.S. add \$.50 per issue ordered. Price includes issue, handling, and shipment by second class or foreign surface mail. Within the U.S., please allow 6-9 weeks to process your order second class. For faster service within the U.S., we'll ship UPS if you add \$1.00 for 1-2 issues and \$.50 for each issue thereafter. We need a street address, not a P.O. Box. Airmail rates: To Canada add \$1.75 per magazine, all other foreign add \$3.00 per magazine. Circle no. 122 on reader service card.

dozens of them) was written by Henry Spencer and posted to Usenet.

system()

Listings of various versions of the `system()` subroutine abound; see [Kernighan and Pike], pages 223-229, and [Kernighan and Ritchie 78b], section 6.

System Calls

Some manuals give the assembly language statements necessary to invoke a given system call (V7 and 4.1 BSD both do this). Later versions of Unix, however, don't do this, presumably because almost no one writes in assembly language anymore. See the section on the kernel (page 51).

tsearch(), tfind(), tdelete(), twalk()

These routines from System V are for working with binary trees; they use algorithms T and D given in section 6.2.2 of [Knuth].

Miscellaneous Utilities

adb

You can get some hints on how `adb` does its work from the `Ptrace(2)` (process trace) man page; note that the details of the system call differ depending on the hardware and the Unix release.

ar

There are various versions of `ar` (file archiver) floating around. The most recent one (in 4 BSD and System V, Release 2) has a public domain equivalent in the `par` (portable archive) and `unpar` programs.

at

[Thomas and Yates] discusses `atrun` and `/usr/spool/at`.

awk

[Aho, Kernighan, and Weinberger] has a particularly concise paragraph in section 5 that explains how `awk` works: it uses `yacc` and `lex`; regular expressions are handled by deterministic finite automata; `awk` builds a parse tree from the program, which is interpreted when actually processing data. [Kernighan and Pike] mentions on page 124 that `awk` uses hashing to implement its associative arrays. See also

the section on regular expressions.

bc

The `BC(1)` man page mentions that `bc` is a preprocessor for `dc` (see below). Various authors note that it is written using `yacc`.

bdiff

[McGilton and Morgan] on page 173 says that `bdiff` uses `split` and `diff` to do its work.

dc

[Morris and Cherry] discusses certain details of `dc`, including how it does arithmetic and the dynamic storage allocation method it uses (described in more detail in [Knowlton]).

df

The `df` (display filesystems) utility calculates the amount of free space on a file system differently in different releases: in V7 it reads the disk's free list, in 4.x BSD it reads the superblock, and in Systems III and V it uses the `ustat(2)` system call. (This information is from Usenet.)

diff

The algorithm used in `diff` was developed (independently) by Harold Stone and by Wayne Hunt and Tom Szymanski; see [Hunt and Szymanski] or [McIlroy and Hunt] for details.

echo

Just about every book on C shows how the `echo` command accesses the command line arguments; [Kernighan and Ritchie 78a] gives three different versions on page 111.

ed

You can get some of the flavor of `ed` from [Kernighan and Plauger 76] and [Kernighan and Plauger 81]; see also the section on regular expressions.

egrep

`egrep` uses an extended version of the normal regular expression algorithms; see the section on regular expressions.

eqn

[Kernighan and Cherry] in sections 5 and 6 supplies various details about the design of `eqn`, including a simplified version of the `yacc` grammar that it

uses.

fgrep

[Whale] gives the source code for a public domain version of `fgrep`; the comments discuss the Knuth-Morris-Pratt string-matching algorithm.

grep

[Kernighan and Pike] discusses the tradeoffs between the different algorithms used in `grep`, `fgrep`, and `egrep`. See the section on regular expressions.

passwd (command)

[Gauthier] provides on page 196 a short explanation of why the `passwd` command must be set-uid root (if this isn't already obvious).

pwd

[Kernighan and Pike] gives hints on pages 51-52 about how `pwd` works.

spell

See [Kernighan and Pike], page 314, [Peterson], and [McIlroy].

Regular Expressions

Regular expressions are used to varying degrees in `awk`, `ed`, `egrep`, Jim Gosling's `emacs`, `grep`, `lex`, `rn` (a "read-news" replacement that was posted to Usenet this summer), `sed`, the shells, `vi` (Berkeley's screen-oriented editor), and `yacc`. While they're very popular under Unix, they aren't used very often in other environments. Rather than discuss regular expressions under each utility that uses them, I've collected all the information here.

Implementation

[Holub] contains source code in C for a version of `grep` that implements full regular expressions. [Kernighan and Plauger 76] and [Kernighan and Plauger 81] give source code to handle slightly different regular expressions in `RATFOR` and `Pascal`, respectively.

Theory

The standard reference is [Aho and Ullman], which cites various other publications on the subject. [Aho and Corasick] may also be a worthwhile paper to read; I haven't seen a copy yet.

References

NEW!

RELOCATABLE Z-80 MACRO ASSEMBLER FROM MITEK

It's a real bargain! Here's why:

- Only \$49.95 plus shipping
- 8080 to Z-80 Source Code Converter
- Generates Microsoft compatible REL files or INTEL compatible hex files
- Compatible with Digital Research macro assemblers MAC & RMAC
- Generates Digital Research compatible SYM files
- Full Zilog mnemonics
- INCLUDE and MACLIB files
- Conditional assembly
- Separate data, program, common and absolute program spaces
- Customize the Macro Assembler to your requirements with installation program
- Over 3 times faster than Microsoft M80 macro assembler
- Z-80 Linker and Library Manager for Microsoft compatible REL files available as a total package with Macro Assembler for only \$95.00
- Manual only is \$15

TO ORDER, CALL TOLL FREE: 1-800-367-5134, ext. 804
For information or technical assistance: 1-808-623-6361

Specify desired 5 1/4" or 8" soft-sectored format. Personal check, cashier's check, money order, VISA, MC, or COD welcomed. Include \$5 for postage and handling.

MITEK

P.O. Box 2151
Honolulu, HI 96805

Z-80 is a trademark of Zilog, Inc. MAC, RMAC, and ZSID are trademarks of Digital Research, Inc. M80 is a trademark of Microsoft Corp.

Circle no. 66 on reader service card.

1 2 3 4 5 **MULTI-TASKING for the IBM AT!** also for the IBM PC or XT with MULTI-JOB™



"MULTI-JOB the most cost effective choice for the user with a need for multi-tasking." PC Age Vol. 3.8

With MULTI-JOB software up to 9 IBM PC DOS compatible programs can be running at the same time. Example, have your communication program running in the background, and still be using your word processing, spreadsheet programs, etc., at the same time! The keyboard and screen can be assigned to any job with a simple keystroke. The remaining jobs will continue to run unattended. With the many different options, MULTI-JOB is a very powerful package.

- * No special hardware is required.
- * Allows priorities to be given between each job.
- * Programs can be run simultaneously or one at a time.
- * 30-day free trial period.

MULTI-JOB	\$159.00
ELECTRONIC DISK	\$ 49.00
SPOOL PROGRAM	\$ 24.00
SET MEMORY UTILITY	\$ 24.00

**B&L COMPUTER CONSULTANTS, 7337 Northview,
Suite B, Boise, ID 83704, (208) 377-8088.**



Dealer's inquiries are welcome. Call or write for a free catalog.

Circle no. 11 on reader service card.

33 KFLOPS

Use your IBM PC, XT or AT to multiply two 128 by 128 matrices at the rate of 33 thousand floating-point operations per second (kflops)! Calculate the mean and standard deviation of 16,384 points of single precision (4 byte) floating-point data in 1.4 seconds (35 kflops). Perform the fast Fourier transform on 1024 points of real data in 6.5 seconds. Near PDP-11/70 performance when running the compute intensive Owen benchmark.

WL FORTH-79

FORTH-79 by WL Computer Systems is a powerful and comprehensive programming system which runs on the IBM PC (and some compatibles) under PC DOS 1.1 or later. If your computer has the 8087 numeric data processing chip (NDP) installed, then this version of FORTH-79 will unleash the awesome floating-point processing power which is present in your system. If you haven't gotten around to installing the 8087 NDP coprocessor in your computer, you can still use WL FORTH to write applications using standard FORTH-79.

System includes editor, memory dump, decompiler, nondestructive stack print-out, screen printer and screen copy utilities. FORTH sources for these utilities are included.

Unlike most other products, the **complete** source is available at a very affordable price.

Package 1 includes FORTH-79 versions with and without 8087 support. Included are screen utilities, 8087 and 8088 FORTH assemblers. \$100

Package 2 includes package 1 plus the assembly language source for the WL FORTH-79 nucleus. \$150

Package 3 includes package 2 plus the WL FORTH-79 source screens used to add the 8087 features to the vocabulary. \$200

Starting FORTH book. \$22

WL Computer Systems
1910 Newman Road
W. Lafayette, IN 47906
(317) 743-8484

Visa and Master Card accepted.

IBM is a trademark of International Business Machines

Circle no. 110 on reader service card.

- [Aho and Corasick] "Efficient String Matching: An Aid to Bibliographic Search," by Alfred V. Aho and M. J. Corasick, in *Communications of the ACM*, 1975, volume 18, pages 333-340.
- [Aho, Kernighan, and Weinberger] "Awk—A Pattern Scanning and Processing Language," second edition (September 1978), by Alfred V. Aho, Brian W. Kernighan, and Peter J. Weinberger, in *UPM*, volume 2. (The first edition of this paper (July 1978) is available in *Software—Practice and Experience*, volume 9.)
- [Aho and Ullman] *Principles of Compiler Design*, by Alfred V. Aho and Jeffrey D. Ullman, Addison-Wesley, 1977.
- [Arnold] "Screen Updating and Cursor Movements Optimization: A Library Package," by Kenneth C. R. C. Arnold, in *UPM for 4.x BSD*, volume 2.
- [Bach and Buroff] "A Multiprocessor Unix System," by Maurice J. Bach and Steven J. Buroff, in *USENIX 1984 Summer Conference Proceedings*, 1984. (See [USENIX Summer '84].)
- [Bourne] *The Unix System*, by Steve R. Bourne, Addison-Wesley, 1982.
- [Carter and Wegman] "Universal Classes of Hash Functions," by Carter and Wegman, in *Proceedings of 9th SIGACT* 1977.
- [Christian] *The Unix Operating System*, by Kaare Christian, John Wiley and Sons, 1983.
- [Comer] *Operating System Design, The XINU Approach*, by Douglas E. Comer, Prentice-Hall, 1984.
- [Fagin, et al.] "Extendible Hashing—A Fast Access Method for Dynamic Files," by Fagin, Nievergelt, Pip-penger, and Strong, in *Transactions on Database Systems*, 1979, volume 4, number 3.
- [Gauthier] *Using the Unix System*, by Richard Gauthier, Reston, 1981.
- [Gimpel] "The Minimization of Spatially Multiplexed Character Sets," by James F. Gimpel, in *Communications of the ACM*, June 1974, volume 17, number 6, pages 315-318.
- [Goodwin] "System V Performance Enhancements," by Ken Goodwin, in *login.*, July 1984, volume 9, number 3, pages 6-11.
- [Hendrix] *The Small-C Handbook*, by J. E. Hendrix, Reston, 1984.
- [Hickman83a] "The Unix Buffer Mechanism—A Detailed Look," by Kipp E. B. Hickman, 1983. (Reprinted in *Writing Unix Device Drivers*; see [Nystrom].)
- [Hickman83b] "The Basics of Writing Unix Device Drivers, with References to the Motorola 68000," by Kipp E. B. Hickman, 1983. (Reprinted in *Writing Unix Device Drivers*; see [Nystrom].)
- [Holt] *Concurrent Euclid, the Unix System, and Tunis*, by R. C. Holt, Addison-Wesley, 1983.
- [Holub] "GREPC—A Unix-like, Generalized, Regular Expression Parser," by Allen Holub, in *Dr. Dobb's Journal*, October 1984, number 96.
- [Hunt and Szymanski] "A Fast Algorithm for Computing Longest Common Subsequences," by J. W. Hunt and Tom G. Szymanski, in *Communications of the ACM*, May 1977.
- [Johnson78a] "LINT—A C Program Checker" (July 26, 1978), by Stephen C. Johnson, in *UPM*, volume 2. (Also available as *CSTR*, number 65.)
- [Johnson78b] "YACC: Yet Another Compiler-Compiler" (1978), by Stephen C. Johnson, in *UPM*, volume 2. (Also available as *CSTR*, number 32.) *Note: the date for this is variously given as 1974, 1975, or 1978; because it contains references to items published in 1978, that seems to be the correct date.*
- [Johnson79] "A Tour Through the Portable C Compiler" (1979), by Stephen C. Johnson, in *UPM*, volume 2.
- [Johnson and Lesk] "Language Development Tools" (December 27, 1977), by Stephen C. Johnson and Michael E. Lesk, in *BSTJ*, July/August 1978, volume 57, number 6, part 2.
- [Johnson and Ritchie 77] "Portability of C Programs and the Unix System" (December 5, 1977), by Stephen C. Johnson and Dennis M. Ritchie, in *BSTJ*, July/August 1978.
- [Johnson and Ritchie 81] "The C Language Calling Sequence" (September 1981), by Stephen C. Johnson and Dennis M. Ritchie, *CSTR*, number 102.
- [Joy] "An Introduction to the C Shell," by William N. Joy, in *UPM for 4.x BSD*, volume 2.
- [Kernighan and Cherry] "A System for Typesetting Mathematics," by Brian W. Kernighan and Lorinda L. Cherry, in *UPM*, volume 2. (An earlier version of this appeared in *Communications of the ACM*, March 1975.)
- [Kernighan and Pike] *The Unix Programming Environment*, by Brian W. Kernighan and Rob Pike, Prentice-Hall, 1984.
- [Kernighan and Plauger 76] *Software Tools*, by Brian W. Kernighan and P. J. Plauger, Addison-Wesley, 1976.
- [Kernighan and Plauger 81] *Software Tools in Pascal*, by Brian W. Kernighan and P. J. Plauger, Addison-Wesley, 1981.
- [Kernighan and Ritchie 78a] *The C Programming Language*, Brian Kernighan and Dennis M. Ritchie, Prentice-Hall, 1978.
- [Kernighan and Ritchie 78b] "Unix Programming—Second Edition" (November 12, 1978), by Brian Kernighan and Dennis M. Ritchie, in *UPM*, volume 2.
- [Knowlton] "A Fast Storage Allocator" (October 1965), by K. C. Knowlton, in *Communications of the ACM*, volume 8, pages 623-625.
- [Knuth] *The Art of Computer Programming—Volume 3, Sorting and Searching*, by Donald E. Knuth, Addison-Wesley, 1973.
- [Lankford] "Unix System V and 4 BSD Performance," by Jeffrey P. Lankford, in *USENIX 1984 Summer Conference Proceedings*, 1984 (See [USENIX Summer '84].)
- [Leffler] "A Detailed Tour Through the /6 Portable C Compiler," (1980?), by Samuel J. Leffler, Department of Computer Engineering, Case Western Reserve University.
- [Leffler, Joy, and Fabry] "4.2 BSD Networking Implementation Notes" (July 1983), by Samuel J. Leffler, William N. Joy, and Robert S. Fabry, in *UPM for 4.2 BSD*, volume 2C. (Also available as UCB CSRG TR/6.)
- [Leffler, Karels, and McKusick] "Measuring and Improving the Performance of 4.2 BSD," by Sam Leffler, Mike Karels, and M. Kirk McKusick, in *USENIX 1984 Sum-*

- mer Conference Proceedings, 1984. (See [USENIX Summer'84].)
- [Lesk and Schmidt] "LEX—A Lexical Analyzer Generator" (1978), by Michael E. Lesk and Eric Schmidt, in *UPM*, volume 2.
- [Lions76] "Unix Operating System Source Code, Level Six" (1976), assembled by John Lions, School of Electrical Engineering, University of New South Wales.
- [Lions77] "A Commentary on the Unix Operating System" (1977), by John Lions, School of Electrical Engineering, University of New South Wales.
- [Lions78] "An Operating System Case Study" (1978), by John Lions in *Operating Systems Review*, July 1978, volume 12, number 3.
- [McGilton and Morgan] *Introducing the Unix System*, by Henry McGilton and Rachel Morgan, McGraw-Hill, 1983.
- [McIlroy] "Development of a Spelling List" (1982), by M. Doug McIlroy, in *IEEE Transactions on Communications*, January 1982.
- [McIlroy and Hunt] "An Algorithm for Differential File Comparison" (1976), by M. Doug McIlroy and J. Wayne Hunt, *CSTR*, number 40 (41?). (Unfortunately, this is obsolete and no longer available from AT&T.)
- [McKusick, Joy, Leffler, and Fabry] "A Fast File System for Unix" (July 27, 1983), by Marshall Kirk McKusick, William N. Joy, Samuel J. Leffler, and Robert S. Fabry, in the *UPM* for 4.2 BSD, volume 2C. (Also available as CSRG TR/7.)
- [McNamara, Vaish, and Bryant] "Writing Device Drivers for XENIX Systems," by Jean McNamara, Parash Vaish, and Richard N. Bryant, in *UniForum Conference Proceedings*, January 1984.
- [Morris and Cherry] "DC—An Interactive Desk Calculator" (November 15, 1978), by Robert Morris and Lorinda L. Cherry, in *UPM*, volume 2.
- [Nowitz] "UUCP Implementation Description" (1978?), by D. A. Nowitz, in *UPM*, volume 2 (of most manuals).
- [Nowitz and Lesk] "A Dial-Up Network of Unix Systems" (1978), by D. A. Nowitz and M. E. Lesk, in *UPM*, volume 2 (of most manuals).
- [Nystrom] *Writing Unix Device Drivers*, by Bob Nystrom, International Technical Seminars, 1984. (Requires a Unix source license; only available by taking the seminar?)
- [Pammett] "A Compiler for the Systems Programming Language—C" (Master's thesis, 1979), by Kevin G. Pammett, Department of Computer Science, University of Waterloo.
- [Peterson] "Computer Programs for Detecting and Correcting Spelling Errors" (1980), by James L. Peterson, in *Communications of the ACM*, December 1980, volume 23, number 12.
- [Ritchie76] "A Tour through the Unix C Compiler" (1976?), by Dennis M. Ritchie, in *UPM*, volume 2.
- [Ritchie78] "The Unix I/O System" (1978), by Dennis M. Ritchie, in *UPM*, volume 2.
- [Ritchie79] "The Evolution of the Unix Timesharing System" (September 1979), by Dennis M. Ritchie, presented at the Symposium on Language Design and Programming Methodology in Sidney, Australia, reprinted in *Microsystems*, October 1984, volume 5, number 10.
- [Ritchie and Thompson] "The Unix Time-Sharing System" (1978), by Dennis M. Ritchie and Ken Thompson, in *BSTJ*, July/August 1978, and *UPM*, volume 2, of most manuals.
- [Schreiner] "p—A Small-C Preprocessor" (1984), by Axel T. Schreiner, in *Dr. Dobbs' Journal*, July 1984, number 93.
- [Thomas and Yates] *A User Guide to the Unix System*, by Rebecca Thomas and Jean Yates, Osborne/McGraw-Hill, 1982.
- [Thompson] "Unix Implementation" (1978), by Ken Thompson, in *BSTJ*, July/August 1978, and *UPM*, volume 2.
- [USENIX Summer'84] *USENIX 1984 Summer Conference Proceedings*, USENIX Association and the Software Tools Users Group, June 12-15, 1984, in Salt Lake City.
- [Vaish and McNamara] "Techniques for Debugging XENIX Device Drivers" (1984), by Parash K. Vaish and Jean Marie McNamara, in *USENIX 1984 Summer Conference Proceedings*, 1984. (See [USENIX Summer

'84].)

- [Wales] "Re: 'panic: iput' message" (October 19, 1983), by Rich Wales (wales@UCLA-LOCUS.ARPA), message number 12791@sri-arpa.UUCP on Usenet.
- [Whale] "fgrep.c—new version" (August 8, 1984), by Geoff Whale (goeff@elecvox.OZ), message number 296@elecvox.OZ on Usenet.

DDJ

Reader Ballot

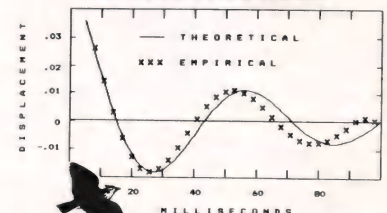
Vote for your favorite feature/article.
Circle Reader Service No. 194.

Graphs without Graphics?

No need for screen graphics. Publishable graphs on your dot matrix printer.

Easy to Use. No programming.
CP/M 80 or 86, MS-DOS, or PC-DOS.
Excellent Manual. Most disk formats.

DataPlotter™



Lark Software™
7 Cedars Road
Caldwell, NJ 07006

Line Graphs & Scatterplots...\$69
Bar Graphs & Pie Charts...\$69
Both for...\$99
(Prices include manual)

Shipping...add \$3
Outside US and Canada...add \$6
Specify which Printer Via M/C
(201) 226-7552

Circle no. 57 on reader service card.

SMALL-C 2.0

for
THE IBM PERSONAL COMPUTER
and
MS-DOS COMPATIBLE SYSTEMS

A large subset of C, packed from argo to xtoi() with features, including

I/O redirection, compilation by parts, peephole optimization, assembly language interface, conditional compilation, switch/case/default, command line support, completion codes, initialization of global variables, data types-char, int, pointers, one-dimensional arrays, and externals

Complete source code for the compiler and libraries plus a 22 page user manual are included.

Requires:

IBM PC assembler, ASM.EXE, or equivalent
IBM PC-DOS 2.0, 2.1 or MS-DOS 2.0
96K memory and a SS Disk Drive

Send \$35 check or money order to

The Coriolis Company
P.O. Box 76
Clinton Corners, NY 12514
(NY residents please add 5% for sales tax.)

Executes sieve benchmark in 35 seconds!

Circle no. 26 on reader service card.

A File Browser Program

by John R. Johnson

Castig about for a useful problem to exercise my new C compiler, I heard about a useful program on a VAX system called "show." This program is similar to the "type" intrinsic command in CP/M, but it allows you to move about at will in the file being displayed. I have often been irritated by "type" when I am looking for a line near the end of a large file and it scrolls past before I can stop the display. It could be a useful utility.

Browse is essentially the front end of a simple editor program. Since it will not change the file, but only display it, a line orientation is adequate. To make moving around easier, the lines in the display should be numbered.

A simple command line parser is required to make the program useful. The ability to repeat a command a set number of times is a desirable option. The parser should accept numeric arguments for either repeat count or line number, depending on the command function.

reading in the file. Should the file be buffered on disk or kept entirely in memory? If it is kept in memory the random access display would be quick. Response time would suffer with a disk buffer. The trade-off is file size. If it is kept entirely in memory the file must be short enough to fit or it will be truncated.

If the file browser has a slow response there is no real reason to use it rather than the editor to examine a file. Any editor will allow most of the functions of the browser. Since it is used primarily for looking over program source code, and I believe programs should be kept as short as possible, I opted for speed. Browse will arbitrarily chop the end off of any file that is too long to fit in the available memory.

The hardware-dependent features are severely restricted. Direct cursor addressing is specifically not required. A string that clears the screen and homes the cursor is used. If your terminal does not support this feature, lo-

Power can be superfluous and features can just be in the way when all you want is the right tool for the task.

There are existing utilities to list files on the printer. A list option in a file browser would be worth including only for those cases where hard copy is desired for only a few lines. If this could be included it would be worth doing.

The most difficult decision involved

cate each instance of the #define constant CLEAR, and replace the call puts(CLEAR) with a call to a function to write a screen full of blanks. Edit the file Browse.h to set the screen and printer page sizes to correspond with your hardware, and that should be all of the installation required.

I do not recommend using I/O redirection with this program. It would put quite a lot of junk into an output file. A minor modification to the list function could allow listing selected portions of

John Johnson, 413 West Sycamore, Carbondale, IL 62901.

the program to an output file. This could be useful if your editor allows you to read in only complete files. You could use Browse to pull a function or two out of a larger file so you could incorporate them into a different program. I chose not to include this capability. It is not useful with my editor.

The various functions in the program are quite straightforward. I have tried to put adequate comments into the source to define each of them.

Browse recognizes several commands for moving around in the file being examined. These commands are explained with the required syntax in the function help(), which is the last function in the source listing. (See the table on page 61 for a list of the Browse commands.) Additional commands could be added by writing the appropriate functions to execute them and adding the command character into the if ... else if ... string in the function command(...). A useful command to add would be a string search capability. This could be patterned after the grep function in the book *Software Tools*, which should be in every serious programmer's library.

At this point in the development, discretion forced a halt before a simple file browser turned into a full-fledged editor. The listing accompanying this article (page 62) was created to be compiled under BDS C version 1.50a. It should convert easily to other C compilers or to other versions of BDS C.

DDJ

(Listing begins on next page)

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 195.

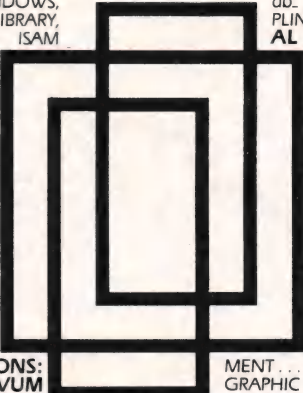
The following is a brief listing of the commands for Browse. The information may also be found in a help screen (at the end of the program listing), which is available to the user on-line. <n> represents any positive integer.

Command	Result
<n>	Redisplay with line <n> in the center
<n> u	Go up <n> lines and redisplay
<n> d	Go down <n> lines and redisplay
b	
Redisplay beginning of file	
e	Redisplay end of file or buffer
<n> t	Reset tab stops to every <n> spaces and redisplay the screen; tab stops default to every four spaces if not set
<n> n	Go down <n> pages (default is one)
<n> p	Go up <n> pages (default is one)
q	Quit and return to operating system
<n1> <n2> 1	List file from <n1> to <n2> on the system list device (line printer)

Table

Once you choose Lattice, our friends will C you through...

LATTICE INC.: LATTICE WINDOWS, CURSES UNIX SCREEN CONTROL LIBRARY, C-FOOD SMORGASBORD, dB-C ISAM COMPATIBLE WITH dBASE II AND III... **LIFEBOAT ASSOCIATES:** FLOAT 87 8087 SUPPORT PACKAGE, HALO GRAPHICS PACKAGE, PANEL SCREEN LIBRARY... **GREENLEAF SOFTWARE:** THE GREENLEAF C FUNCTIONS... **C SOURCE:** BASIC, C C FUNCTIONS FOR BASIC USER... **SOFTCRAFT:** RETRIEVE ISAM FILE SYSTEM, RETRIEVE ISAM NETWORK FILE SYSTEM... **BLAISE COMPUTING:** TOOLS, TOOLS2, VIEW/MANAGER, SCREEN PACKAGE... **MORNING STAR SYSTEMS:** PROLIBRARY, PRO-SCREEN... **CREATIVE SOLUTIONS:** WINDOWS FOR C... **NOVUM ORGANUM:** C POWERS PACKS, MATHEMATICS POWER PACKS, ADVANCED POWER PACKS, DATABASE POWER PACKS, TELECOMMUNICATIONS POWER PACKS W/ SOURCE... **PHACT ASSOCIATES:** PHACT ISAM LIBRARY... **RAIMA CORPORATION:**



db. VISTA DBMS... **PHOENIX:** PLINK86, PFIX86... **RELATIONAL DATABASE SYSTEMS:** C-ISAM FILE ACCESS METHOD... **MINDBANK:** V-FILE VIRTUAL MEMORY/FILE SYSTEM... **HUNTER & READY:** VRTX C INTERFACE LIBRARY... **GRAPHIC SOFTWARE SYSTEMS:** GSS DRIVERS, GSS TOOLKIT KERNEL SYSTEM... **OPT-TECH DATA PROCESSING:** OPT-TECH SORT... **ACCU DATA SOFTWARE:** C-TREE ISAM, C-SORT SORT... **TRIO SYSTEMS:** C-INDEX+ ISAM... **COMPU CRAFT:** c VIEW FORMS/WINDOW/MANAGEMENT... **SCIENTIFIC ENDEAVORS:** GRAPHIC PRESENTATION SCIENTIFIC GRAPHICS... **LEMMA SYSTEMS, INC.:** C LIBRARY... **ESSENTIAL SOFTWARE, INC.:** C UTILITY LIBRARY... **SOFTWARE LABS:** C UTILITIES PACKAGE... **FAIRCOM:** C-tree BY FAIRCOM ISAM WITH SOURCE

Contact Lattice to learn how we can help your C program development.



LATTICE

P.O. Box 3072
Glen Ellyn, IL 60138
312/858-7950
TWX 910-291-2190


```
/*
 * Program Browse.c
 * by John R. Johnson
 * Version 1.01      Dec 7, 1983
 *
 * Copyright 1983
 * by John R. Johnson
 * All rights reserved.
 * Permission is granted for
 * unlimited personal,
 * non-commercial use only.
 *
 * Address queries to:
 *
 * John R. Johnson
 * Professional Microware, Inc.
 * P. O. Box 200
 * Carbondale, Illinois, 62903
 *
 * Reasonable telephone queries
 * will be answered if you call
 * at 618-529-2717. Make sure
 * the time is between 9 AM and
 * 5 PM Central time.
 * I regret that I cannot return
 * long distance calls except
 * collect. Thank You.
 */
```

```
#include <bdscio.h>
/* Leor Zolman's definitions */
#include <browse.h>
/* browse definitions */
```

```
main(argc,argv)
int argc;
char **argv;
{
    int tabstop;
    /* tab stops for display */
    int lincnt;
    /* number of lines in buffer */
    int offset;
    /* window width / 2 */
    int curlin;
```

```
/*
 * active line number
 * ( center of window )
 */
```

```
int *lines;
char *max;
/*
 * array of pointers to strings.
 * This array is the line index
 * for the file. max is the
 * maximum useable address for
 * buffer. The buffer is
 * allocated but not accessible
 * except through this array.
 * The array lines[lincnt] is a
 * trick to get an array of
```

```
 * dynamically assigned length.
 * It is located by makbuf()
 * and created by filbuff( ).
 */
```

```
char inbuf[BUFSIZ];
/* file input buffer */
char filename[18];
/* filename buffer */
char cline[135];
/* command line buffer */
```

```
puts(CLEAR);
printf("\nBROWSE copyright ");
printf("1983 by John R.");
printf(" Johnson\n");
```

```
tabstop=4;
/* default value for tabs */
```

```
/*
 * get the file name, if it is
 * not given on the command
 * line prompt the user.
 */
```

```
if ((argc>1) &&
    (strlen(argv[1])<18))
{
    strcpy(filename, argv[1]);
}
```

```
else
{
    while(1)
    {
        puts("Enter file name > ");
        if (getline(filename, 18))
            break;
    }
}
```

```
/*
 * open the file requested
 */
```

```
if (fopen(filename,inbuf)==ERROR)
{
    error("\nfile %s not found",
        filename);
    exit();
}
```

```
/*
 * create the arrays
 */
```

```
lines = makebuf(&max);
if (lines==0)
{
    error("\ncouldn't create
        buffer for %s",filename);
    fclose(inbuf);
    exit();
}
```



```

}

/*
 * read the file into the arrays
 */

if ((lincnt=
    fillbuf(lines,max,inbuf)) == 0)
{
    error("\ntrouble reading %s",
          filename);
    fclose(inbuf);
    exit();
}

/*
 * close the file so we don't
 * do anything to it.
 */

fclose(inbuf);

/*
 * initialize the display
 */

offset = setoff();
curlin = offset;

/*
 * display the screen and parse
 * and interpret commands.
 * Notice the next current line
 * is returned from the command
 * parse routine.
 */

while((curlin=command(filename,
    curlin,cline,offset,lines,
    lincnt,&tabstop))
    ;

/*
 * clear the screen
 */

puts(CLEAR);

} /* end of the main function */

/* special functions used */

/*
 * int error(format, arg)
 * char *format, *arg;
 *
 * ring the bell and print an
 * error message formatted as
 * for the printf function.
 */

int error(format,arg)
char *format,*arg;

{
    puts(BELL);
    printf(format,arg);
}

```

```

}

/*
 * int setoff()
 *
 * set the display offset
 */

int setoff()
{
    return ((SCRHT-1)/2);
}

/*
 * display(keylin, offset, lines,
 *          last, tab, f)
 * int keylin, offset, lines[];
 * int last, tab, f;
 *
 * display offset lines before and
 * after the current line. Stop
 * when screen is full and wait
 * for further commands. Do not
 * alter the display if current
 * line is negative. A negative
 * value is used to keep from
 * writing over the help screen
 * when it is displayed.
 */

display(keylin,offset,lines,last,
        tab,f)
int keylin,offset,lines[],last,tab;
char *f;
{
    int j, lone, ltwo;

    /*
     * if current line is negative,
     * omit display update
     */

    if (keylin<0)
        return;

    /*
     * don't try to display more
     * lines than there are.
     */

    if ((keylin+offset)>last)
        keylin=last-offset;

    /*
     * don't try to display before
     * the beginning of the file.
     */

    if ((keylin-offset)<0)
        keylin=offset;

    /*
     * if the entire file fits on
     * the screen, just display the
     * entire file. Otherwise just
     * display what fits, so it the
     * current line stays in the
     * middle of the screen.
     */

    if (last<(offset+offset+1))

```

(Continued on page 66)

*C Is The Language.
Lifeboat Is The Source.*



*Lifeboat.TM
The Leading Source And Authority For Serious Software.
1-800-847-7078.*

In NY State: 212-860-0300

Serious Software For The C Programmer From Lifeboat.™

Lattice[®] C Compiler: *The serious software developer's first choice.*

Selected for use by IBM,[®] Texas Instruments, Wang,[®] MicroPro,[®] Ashton-Tate,[™] IUS/Sorcim,[®] Microsoft[®] and Lotus[™] to name a few of the many. Why?

Lattice C is clearly the finest 16 bit C compiler available today.

- Renowned for speed and code quality.
- Fully compatible with the C standards set forth by Kernighan and Ritchie.
- Four memory model options offer you unsurpassed control and versatility.
- Superior quality documentation.
- Now includes automatic sensing and use of the 8087 chip.
- Widest selection of supporting add-on packages.

Halo[™]: *A graphics development package rapidly emerging as the industry standard.*

- 140 graphics commands including plot, line, arc, box circle and ellipse primitives, bar and pie charts; pattern fill and dithering commands.
- New: multiple viewports and "stroke text" for angling, scaling and filling text.

C Food Smorgasbord[™]: *This beautifully written collection of C functions is a valuable time saver.*

- Library includes a binary coded decimal arithmetic package, level 0 I/O functions, a terminal independence package, IBM PC ROM BIOS access functions and much more.

Pmate[™]: *The premier editor for the programming professional.*

Pmate is a full screen editor with its own powerful macro command language:

- Perform on screen row and column arithmetic, alphabetize lists, translate code from one language to another, call up other macros.
- Customize Pmate almost any way you like.
- Contains 10 auxiliary buffers for storage of macros, text, subroutines.
- An "undo" feature allows the programmer to retrieve whole series of deleted items.

Additional C Tools

Available From Lifeboat:

Panel[™]: Screen formatter and data entry aid.

Lattice Windows[™]: Windowing utility; create "Virtual Screens."

Plink-86[™]: The popular linker; includes extensive overlay capabilities.

Pfix86[™]: Dynamic debugging utility.

Pfix86 Plus[™]: Symbolic debugger with capacity to debug overlays.

Btrieve[™]: Database record access/retrieval library.

Phact: Multikeyed ISAM C-Function library.

Fabs: Fast access B-tree database function library.

Autosort: Fast sort/merge utility.

ES/P: 'C' program entry with automatic syntax checking and formatting.

Greenleaf Functions[™]: Library of over 200 popular C functions.

And much more.

YES! Please rush me the latest FREE Lifeboat[™] catalog of C products.

Company
Name _____

Business
Phone _____

Name _____

Title _____

Address _____

City _____

State _____

Zip _____

Please check the category where Lifeboat can best help you:

☐ Software development

☐ Corporate

☐ Education

☐ Dealer/distributor

☐ Government

☐ Other _____

Call Direct: 1-800-847-7078 (In NY State: 212-860-0300)

**Return coupon to: Lifeboat Associates[™]
1651 Third Avenue, New York, NY 10128.**

Catalog 25 1984
C Programming Tools




```

{
    lone=0;
    ltwo=last;
}
else
{
    lone=keylin-offset;
    ltwo=keylin+offset;
}

/*
 * do the display of the
 * lines we chose and then
 * print the file name at the
 * bottom of the screen.
 */
puts(CLEAR);
for (j=lone;j<(ltwo);++j)
    outlin(j,lines[j],tab);
printf("file: %s > ",f);
}

* int n, t;
* char *str;
*
* output a line to the terminal
* expanding tabs by t. To take
* care of terminals with fixed
* tab stops. Truncate displayed
* line to screen width to prevent
* uncontrolled scrolls.
*/

int outlin(n,str,t)
int n,t;
char *str;
{
    char c;
    int col,k;

    k=0;
    printf("%3d: ",n);
    /* print the line number */

    /*
     * now output the line
     */

    for (col=5;col<(SCRWID-6);)
    {
        /*
         * expand tabs for terminals
         * that don't support tab
         * setting to any widths.
         */

        if ((c=str[k++])=='\t')
        {
            do
                /* at least one blank */

```

```

        {
            putchar(' ');
            col++;
        } while((col%t)&&
                (col<(SCRWID-6)));
    }

    /*
     * quit at the new line
     */
    else if ( c=='\n' )
    {
        break;
    }

    /*
     * for all others,
     * just put the char
     */

    else
    {
        putchar(c);
        col++;
    }
}

/*
 * always end with a new line
 */
putchar('\n');
}

/*
 * int makebuf(pmax)
 * char **pmax;
 *
 * allocate all of free memory for
 * a buffer for the lines array
 * and the text buffer. Arg is a
 * pointer to max so that max can
 * be set to the last available
 * memory location. Return the
 * pointer gotten from alloc(...)
 * for the value of lines.
 */

int makebuf(pmax)
char **pmax;
{
    char *here;
    unsigned templ,temp2;

    templ=endext();
    /* first memory location */
    temp2=topofmem()-3000;

    /* last free memory */
    here=alloc(temp2-templ);
    /* allocate it */
    if (here==0)

```

(Continued on page 68)

C Programmers: Program three times faster with *Instant-C*[™]

Instant-C[™] is an optimizing **interpreter** for C that makes programming three or more times faster. It eliminates the time wasted by compilers. Many repetitive tasks are automated to make programming less tedious.

- Two seconds elapsed time from completion of editing to execution.
- **Symbolic debugging**; single step by statement.
- Compiled execution speed; 40 times faster than interpreted Basic.
- Full-screen editor integrated with compiler; compile errors set cursor to trouble spot.
- Directly generates .EXE or .CMD files.
- Follows K & R—works with existing programs. Comprehensive standard C library with source.
- Integrated package; nothing else needed.
- Works under PC-DOS*, MS-DOS*, CP/M-86*.

More productivity, less frustration, better programs.

Instant-C[™] is \$500. Call or write for more info.

**Rational
Systems, Inc.**

(617) 653-6194
P.O. Box 480
Natick, Mass. 01760

Trademarks: MS-DOS (Microsoft Corp.), PC-DOS (IBM), CP/M-86 (Digital Research, Inc.), Instant-C (Rational Systems, Inc.)

Circle no. 79 on reader service card.

Turbo + PC Tools = Programs

Tools for Turbo Pascal[™] on the IBM PC

Window Management = menus, help files . . .

- Unlimited windows
- Window overlay & recall
- Cursor save & jump
- Access all colors & chars
- Window Compiler/Librarian manages window files

Graphics Drawing = HiRes plotting power!

- Ellipses, polygons & more
- Region fill and clear

String Formula Evaluator = easy calculation

- 22 functions with nesting and implicit multiplication
- Won't bomb on overflow or division by zero

System Check and Control = max flexibility!

- Time & date access
- Get disk types & room
- Get & set default drive
- I/O information

All this for only \$39.95* . . . Incredible!

You get 321K of *source* code on a double-sided disk and a 35 page manual. For single-sided drives add \$2. Works with DOS 2.0, Turbo 2.0.

*Please include \$2 for postage and handling (\$4 if outside of USA). Californians add 6%.

Paragon Courseware
4954 Sun Valley Road
Del Mar, CA 92014
(619) 481-1477

Turbo Pascal is a trademark of Borland International

Circle no. 69 on reader service card.

Subscribe to Dr. Dobb's Journal For The Holidays And Give Twelve Gifts in One!

For Christmas, treat a friend (or yourself!) to a full year (12 issues) of Dr. Dobb's Journal, delivered monthly to the doorstep, and save \$5 off the newsstand price.

Better yet, subscribe for 2 years and save \$13!

Every month, you'll find articles on Small-C, FORTH, CP/M, S-100, Compiler optimization, Concurrent Programming and more.

Dr. Dobb's Journal, P.O. Box 27809, San Diego, CA 92128

Yes! Please start a subscription for
_____ 1 yr. \$25 _____ 2 yrs. \$47 for:

Name _____
Address _____

_____ Zip _____

_____ Charge my Visa, MasterCard,

_____ Please bill me later

_____ I enclose a check/money order

Name _____
Address _____

_____ Zip _____

Credit Card _____ Exp. date _____

Account No. _____

Signature _____


```

    return (0);
    *pmax = temp2;
    return (here);
}

/*
 * int fillbuf(lines, max, inbuf)
 * int lines[];
 * char *max;
 * struct _buf *inbuf;
 *
 * Fill the string arrays allocated
 * by makbuf(). As each line is
 * placed into the text buffer,
 * enter the corresponding pointer
 * into the array lines[]. Notice
 * the text buffer builds down from
 * top of free memory while the
 * array lines[] builds up from the
 * bottom. Filling stops on end of
 * file or when the array meets the
 * text buffer.
 * Return the number of lines read.
 */

```

```

int fillbuf(lines,max,inbuf)
int lines[];
char *max;
struct _buf *inbuf;
{
    int count;
    /* line counter */
    char *bufptr,*buflin;
    /* buffer ptrs */
    char tbuf[100];
    /* temp input buffer */

    bufptr = max;
    /* start of buffer */
    *bufptr-- = '\0';
    /* put in the null */
    *bufptr-- = '\0';
    buflin = bufptr - 8;

    if (lines > buflin)
        return (0);
    strcpy(buflin,"<Start>\n");
    lines[0] = buflin;
    bufptr = buflin;
    count=1;
    while (&lines[count] < bufptr)
    {
        /*
         * get a line into the buffer
         * and place the pointer in
         * lines[] array for access.
         */
        if (fgets(tbuf, inbuf))
        {
            buflin = bufptr -
                strlen(tbuf) - 1;
            if (&lines[count] > buflin)
                break;
            *--bufptr = '\0';

```

```

        lines[count] = buflin;
        bufptr = buflin;
        strcpy(buflin, tbuf);
        count++;
    }
    else
    {
        break;
    }
}
buflin = bufptr - 8;
if (&lines[count] < buflin)
{
    /*
     * put on the end of file
     * marker if we got there.
     */

    strcpy(buflin,"<End>\n");
    lines[count] = buflin;
    *--bufptr = '\0';
    *--bufptr = '\0';
    ++count;
}
return (count);
}

/*
 * int command(f, active, cmd,
 *             offset, lines, lincnt, t)
 * int active, offset, lines[];
 * int lincnt, *t;
 * char cmd, *f;
 *
 * command interpreter and display
 * handler. Parses the command
 * line and executes the correct
 * function to execute the command.
 * Additional commands can be added
 * into the if ... else if ... else
 * construction in the parser.
 */

int command(f,active,cmd,offset,
            lines,lincnt,t)
int active,offset,lines[];
int lincnt,*t;
char cmd[],*f;
{
    int length,first,second,temp;
    char key;

    display(active,offset,lines,
            lincnt,*t,f);

    /*
     * If display is suppressed,
     * then activate it.
     */
    if (active<0)
        active=-(active);

```

(Continued on page 70)



Eco-C Compiler

Release 3.0

We think Rel. 3.0 of the Eco-C Compiler is the fastest full C available for the Z80 environment. Consider the evidence:

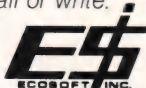
Benchmarks* (Seconds)

Benchmark	Eco-C	Aztec	Q/C
Seive	29	33	40
Fib	75	125	99
Deref	19	CNC	31
Matmult	42	115	N/A

*Times courtesy of Dr. David Clark
CNC - Could Not Compile
N/A - Does not support floating point

We've also expanded the library (120 functions), the user's manual and compile-time switches (including multiple non-fatal error messages). The price is still \$250.00 and includes Microsoft's MACRO 80. As an option, we will supply Eco-C with the SLR Systems assembler - linker - librarian for \$295.00 (up to six times faster than MACRO 80).

For additional information,
call or write:



ECOSOFT, INC. (317) 255-6476
6413 N. College Ave. • Indianapolis, Indiana 46220



NEW RELEASE

Circle no. 35 on reader service card.

"Ouvrez les fenêtres!"*

Introducing **MATIS™**, the powerful new
developmental system from France.

A complete and meticulously detailed program
to make a programmer's work easier, faster, and...
but of course... better.

- ☐ Window Management Systems ☐ Screen Generator ☐ Expanded Basic Commands ☐ Can be accessed from other languages ☐ 100% Assembler ☐ Automatic Scrolling in Windows ☐ Virtual Page larger than screen (up to 65534 rows x 65534 columns) ☐ Save or Print Pages ☐ MS-DOS ☐ 170 Page Manual (In English Mon Ami!)
- ☐ Only \$150.

ORDER BY MAIL—WRITE OR CALL FOR COMPLETE DESCRIPTION
No license fee.

Softway, Inc.

500 Sutter Street • Suite 222— M • San Francisco, CA 94102
Tel: (415) 397-4666 Telex: 880857

*"Open the windows!"

Circle no. 92 on reader service card.

Of course, POWER!™ saves your Bad Disk.

NOW! WINDOWS FOR IBM!



It also does
54 other things to
keep your disk in line.

EVERYTHING YOU ALWAYS WANTED TO DO, BUT WERE AFRAID TO TRY

Unlike some utility programs that are a headache to use, POWER! is engineered to spoil you with 55 features, simple and uniform commands, and utter simplicity of use. POWER! automatically alphabetizes and numbers your files. You select by the number and never type file names again. Need to [COPY], [RENAME], [ERASE], or [RUN] programs? Just type in their menu number! POWER! also locks out your disk's bad sectors [TEST] without destroying files—a critical difference from other utilities that search and destroy, without informing you what they've done, leaving you to wonder why your programs won't run. (And POWER! still has 50 commands to go!)

POWER! ONE PROGRAM DOES IT ALL!

You may own a few utility programs for your computer housekeeping, each with its own commands to memorize. POWER! has all the programs rolled into one 16K integrated package, so you do things you've never tried before—every day. Save sensitive data from prying eyes with [PASS] word protect, move a block of memory [MOVE], look for data [SEARCH] or compare files [CHECK]. POWER! also makes easy work of patching, [DISPLAY/SUBSTITUTE], customizing software [LOAD/SAVE]. Among the other commands are [SIZE], [STAT] [LOG], [DUMP], [TYPE], [JUMP], [FILL], [SET], and the CP/M version lets you restore erased files—even when you don't remember the filename—at a flick of the POWER! [RECLAIM] command. (Still 31 commands to go!)

POWER! NOW FOR IBM's PC-DOS AS WELL AS CP/M

We first developed POWER! for CP/M two years ago, and a stack of testimonials from FORD to XEROX testify to its excellence. For IBM-PC™ users, special features like managing sub-directories, [CHANGE], and a separate creation of up to 8 simultaneous, on-screen [WINDOWS] have been added.

MONEY-BACK GUARANTEE AND A 10 DAY TRIAL

POWER! has the Seal of Approval from the Professional Software Programmers Association, and you, too, must be happy with POWER!—or your money back! For only \$169 you can now really be in control of your computer. Call Computing! at (415) 567-1634, or your local dealer. For IBM-PC or any CP/M machine. Please specify disk format.

The company that earns its exclamation point.

COMPUTING!

2519H Greenwich, San Francisco, CA 94123

TO ORDER CALL 800 TOLLFREE
800-428-7825 Extension 96H
In CA: 800-428-7824 Extension 96H

IBM and IBM-PC are registered trademarks of
International Business Machines Corporation.

Circle no. 25 on reader service card.


```
/* get a command line */
length=getline(cmd, 132);

/*
 * if command line is null,
 * then do nothing more.
 */
if (length == 0)
    return (active);

/*
 * Set the key to the first
 * alpha in command.
 */
key=letter(cmd,length);

/*
 * Get up to two numbers from
 * the command line.
 */
first=numone(cmd,length);
second=numtwo(cmd,length);

/*
 * No key found means first is
 * the desired current line.
 */
if (key == 0)
{
    if (first<offset)
        return (offset);
    if ((first+offset)>lincnt)
        return (lincnt-offset);
    return (first);
}

/*
 * force the key to lower case.
 */
key = tolower(key);

/*
 * if key is u
 * move up n lines.
 * (default 1)
 */
if (key == 'u')
{
    if (first)
    {
        if ((active-first)
            < offset)
            return (offset);
        else
            return (active-first);
    }
    else
    {
        if (active > offset)
            return (--active);
        else
            return (offset);
    }
}
```

```

}

/*
 * if key is d
 * move down n lines.
 * (default 1)
 */
if (key == 'd')
{
    if (first)
    {
        if ((active+first) >
            (lincnt-offset))
            return (lincnt-offset);
        else
            return (active+first);
    }
    else
    {
        if (active <
            (lincnt-offset))
            return (++active);
        else
            return (lincnt-offset);
    }
}

/*
 * if key is b
 * move to beginning of file.
 */
if (key == 'b')
    return (offset);

/*
 * if key is e
 * move to end of file.
 */
if (key == 'e')
    return (lincnt-offset);

/*
 * if key is t
 * set tabs to n (default 4)
 */
if (key == 't')
{
    if ((first<=0)|| (first>20))
        first=4;
    *t=first;
    return (active);
}

/*
 * if key is p
 * n'th previous page.
 * (default 1)
 */
if (key == 'p')
{
```

(Continued on page 72)

C UTILITY LIBRARY

The **C UTILITY LIBRARY** is a set of **200+** functions designed specifically for the PC software developer. Use of the Library will speed up your development efforts and improve the quality of your work.

- **BEST SCREEN HANDLING AVAILABLE**
- **WINDOW MANAGEMENT, COLOR GRAPHICS**
- **DOS 2 DIRECTORIES, COMMUNICATIONS**
- **KEYBOARD, PRINTER, TIME/DATE**
- **EXECUTE PROGRAMS, BATCH FILES**
- **STRINGS, BIOS, AND MUCH MORE**
- **ALL SOURCE INCLUDED—NO ROYALTIES**

Available for Microsoft/Lattice \$149, Computer Innovations \$149, Mark Williams \$149, DeSmet \$99. Add \$3 shipping. N.J. residents add 6% sales tax. Visa, MC, checks—10 days to clear. Order direct or through your dealer. Dealer/Distributor inquiries welcome.

ESSENTIAL SOFTWARE, INC.
(914) 762-6605
P.O. Box 1003
Maplewood, N.J. 07040

Circle no. 36 on reader service card.

\$3.00 C Compiler

Due to popular demand Dr. Dobb's Journal has reprinted Ron Cain's C compiler from our sold-out 1980 issues #45 and #48. The reprint includes "A Small C Compiler for the 8080s" and "Runtime Library for the Small C Compiler" for only \$3.00, postage included. To Order:

Enclose \$3.00 for each copy with this coupon and send to:
Dr. Dobb's Journal, 2464 Embarcadero Way,
Palo Alto, CA 94303.

Please send _____ copy(ies) to:

name _____

address _____

city _____ state _____ zip _____

All reprint orders must be prepaid.

98

Circle no. 126 on reader service card.

C Source Code

RED

Full Screen Text Editor

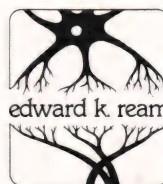
New

Versions for IBM PC and Kaypro.
Distributed on 5¼ disk formats.

- RED is fast! RED uses all of your terminal's special functions for best screen response. RED handles files as large as your disk automatically with a minimum of disk traffic.
- RED is easy to use for writers or programmers. It uses plain English commands.
- RED comes with complete source code in standard C. You choose how your editor will work. RED's code is truly portable: RED has been ported to mainframes, minis and micros.
- RED comes with a Reference Card and a Reference Manual that provides everything you need to use RED immediately.
- RED is unconditionally guaranteed. If for any reason you are not satisfied with RED your money will be refunded promptly.

RED: \$95

Manual: \$10



Call or write today for more information:
Edward K. Ream
1850 Summit Avenue
Madison, WI 53705
(608) 231-2952

To order:

Either the BDS C compiler or the Aztec CII compiler is required for CP/M 80 systems. Digital Research C compiler v1.1 is required for CP/M 68K systems. No compiler is required for IBM or Kaypro systems.

Specify both the machine desired (IBM, Kaypro or CP/M) and the disk format desired (8 inch single density CP/M or exact type of 5¼ inch disk).

Send a check or money order for \$95 (\$105 U.S. for foreign orders). Sorry, I do NOT accept phone, credit card, or COD orders. Please do not send purchase orders unless a check is included. Your order will be mailed to you within one week.

Dealer inquiries invited.


```

if (first)
{
    if ((temp=2*offset*first)
        > active)
        return (offset);
    else
        return (active-temp);
}
else
{
    if ((temp=2*offset)
        > active)
        return (offset);
    else
        return (active-temp);
}
}

/*
 * if key is n
 * n'th next page (default 1).
 */
if (key == 'n')
{
    if (first)
    {
        temp=first*(offset+offset);
        if ((active+temp)
            > (lincnt-offset))
            return (lincnt-offset);
        else
            return (active+temp);
    }
    else
    {
        temp=offset+offset;
        if ((active+temp)
            > (lincnt-offset))
            return (lincnt-offset);
        else
            return (active+temp);
    }
}

/*
 * if key is q
 * quit and exit to system.
 */
if (key == 'q')
    return (0);

/*
 * if key is l
 * list from first to second
 * lines on printer.
 */
if (key == 'l')
{
    lister(first, second, lincnt, lines,
        *t,f);
    return (active);
}

```

```

/*
 * default, illegal command.
 */
/*
 * display help screen for
 * everything else.
 */
help();
return -(active);
}

/*
 * int letter(str,len)
 * char *str;
 * int len;
 *
 * returns the first alphabetic
 * character in string str
 * returns null if there are no
 * alphas in the string.
 */

char letter(str,len)
char *str;
int len;
{
    char c,*cptr;
    int k;

    cptr=str;
    k=0;
    while((c=cptr[k++]))
        if (isalpha(c))
            break;
    return (c);
}

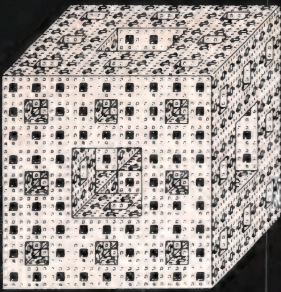
/*
 * numone(str, len)
 * char *str;
 * int len;
 *
 * returns the integer value of
 * the first decimal digit string
 * encountered in string str.
 * Returns zero if no digit string
 * is found.
 */

int numone(str,len)
char *str;
int len;
{
    char c, *cptr;
    int k;

    cptr=str;
    k=0;
    while ((c=cptr[k]))
    {

```

(Continued on page 74)



WALTZ LISP^(TM)

The one and only **adult** Lisp system for CP/M users.

Waltz Lisp is a very powerful and complete implementation of the Lisp programming language. It includes features previously available only in large Lisp systems. In fact, Waltz is substantially compatible with *Franz* (the Lisp running under *Unix*), and is similar to *MacLisp*. Waltz is perfect for Artificial Intelligence programming. It is also most suitable for general applications.

Much faster than other microcomputer Lisps. • Long integers (up to 611 digits). Selectable radix • True dynamic character strings. Full string operations including fast matching/extraction. • Flexibly implemented random file access. • Binary files. • Standard CP/M devices. • Access to disk directories. • Functions of type lambda (expr), lambda (fexpr), lexpr, macro. • Splicing and non-splicing character macros. • User control over all aspects of the interpreter. • Built-in prettyprinting and formatting facilities. • Complete set of error handling and debugging functions including user programmable processing of undefined function references. • Virtual function definitions. • Optional automatic loading of initialization file. • Powerful CP/M command line parsing. • Fast sorting/merging using user defined comparison predicates. • Full suite of mapping functions, iterators, etc. • Assembly language interface. • Over 250 functions in total. • The best documentation ever produced for a micro Lisp (300+ full size pages, hundreds of illustrative examples).

Waltz Lisp requires CP/M 2.2, Z80 and 48K RAM (more recommended). All common 5" and 8" disk formats available.

PROCODE^(TM)
INTERNATIONAL

15930 SW Colony Pl.
Portland, OR 97224

Unix* Bell Laboratories.
CP/M* Digital Research Corp.

Version 4.4

(Now includes Tiny Prolog
written in Waltz Lisp.)

*Manual only: \$30 (refundable with order). All foreign orders: add \$5 for surface mail, \$20 for airmail. COD add \$3. Apple CP/M and hard sector formats add \$15.

Call free **1-800-LIP-4000** Dept. #11
In Oregon and outside USA call 1-503-684-3000

\$169*

Circle no. 73 on reader service card.

AVAILABLE BACK ISSUES

1982	1983	1984
No. 64—Feb.	No. 76—Feb.	No. 87—Jan.
No. 66—April	No. 77—March	No. 88—Feb.
No. 68—June	No. 78—April	No. 89—March
No. 69—July	No. 79—May	No. 90—April
No. 70—Aug.	No. 80—June	No. 91—May
No. 71—Sept.	No. 81—July	No. 92—June
No. 72—Oct.	No. 82—Aug.	No. 93—July
No. 73—Nov.	No. 83—Sept.	No. 94—Aug.
	No. 84—Oct.	No. 95—Sept.
	No. 85—Nov.	No. 96—Oct.
	No. 86—Dec.	No. 97—Nov.

TO ORDER: send \$3.50 per issue to: Dr. Dobb's Journal,
2464 Embarcadero Way, Palo Alto, CA 94303.

Name _____

Address _____

City _____

State _____

Zip _____

98

Circle no. 122 on reader service card.

LISP FOR THE IBM PERSONAL COMPUTER.

THE PREMIER LANGUAGE
OF ARTIFICIAL
INTELLIGENCE FOR
YOUR IBM PC.

DATA TYPES

Lists and Symbols
Unlimited Precision Integers
Floating Point Numbers
Character Strings
Multidimensional Arrays
Files
Machine Language Code

MEMORY MANAGEMENT

Full Memory Space Supported
Dynamic Allocation
Compacting Garbage Collector

FUNCTION TYPES

EXPR/FEXPR/MACRO
Machine Language Primitives
Over 190 Primitive Functions

IO SUPPORT

Multiple Display Windows
Cursor Control
All Function Keys Supported
Read and Splice Macros
Disk Files

POWERFUL ERROR RECOVERY

8087 SUPPORT

COLOR GRAPHICS

LISP LIBRARY

Structured Programming Macros
Editor and Formatter
Package Support
Debugging Functions
.OBJ File Loader

RUNS UNDER PC-DOS 1.1 or 2.0

IQLISP

5 1/4" Diskette
and Manual _____ \$175.00
Manual Only _____ \$ 30.00

Integral Quality

P.O. Box 31970
Seattle, Washington 98103-0070
(206) 527-2918

Washington State residents add sales tax.
VISA and MASTERCARD accepted.
Shipping included for prepaid orders.

Circle no. 49 on reader service card.


```

        if (isdigit(c))
            break;
        ++k;
    }
    if (c)
        return (atoi(cpstr+k));
    return (0);
}

/*
 * int numtwo(str, len)
 * char *str;
 * int len;
 *
 * returns the integer value of
 * the second decimal digit string
 * found in string str.
 * Returns zero if there is no
 * second digit string in str.
 */

int numtwo(str,len)
char *str;
int len;
{
    char c,*cpstr;
    int k,n;

    cpstr=str;
    k=0;
    while ((c=cpstr[k]))
    {
        if (isdigit(c))
            break;
        k++;
    }
    if (c==0)
        return (0);
    cpstr=nondigit(cpstr+k);
    k=cpstr-str;
    return (numone(cpstr,len-k));
}

/*
 * char *nondigit(ptr)
 * char *ptr;
 *
 * advance ptr to the first
 * position that is not a digit
 * and return the new value of
 * the ptr.
 */

char *nondigit(ptr)
char *ptr;
{
    char c,*cpstr;

    cpstr=ptr;
    while(1)
    {
        c=*cpstr;

```

```

        if (isdigit(c)==FALSE)
            return (cpstr);
        ++cpstr;
    }
}

/*
 * lister(here, tohere, topcnt,
 *         lines, tab, f)
 * int here, tohere, topcnt;
 * int lines[], tab;
 * char *f;
 *
 * Print the lines of the file
 * from here tohere on the system
 * list device using CP/M list
 * driver. If here is greater
 * than tohere roll around to
 * line 0 when the end of file
 * is reached and continue list.
 * The listing should be paginated
 * according to the values in
 * brouse.h for form width and
 * length. Each line should be
 * numbered. Expand tabs by tab.
 * Print the filename and a page
 * number at the top of each page.
 */

lister(here, tohere, topcnt,
        lines, tab, f)
int here, tohere, topcnt;
int lines[], tab;
char *f;
{
    char c,*cpstr;
    int col,row,i,j,page;

    if ((tohere<=0) ||
        (tohere>=topcnt))
        tohere=topcnt-1;
    if ((here<=0) ||
        (here>=topcnt))
        here = 0;

    i=here;
    page=row=1;
    col=0;

    while (1)
    {
        fprintf(2,"\\filename: %s"
                ,f);
        fprintf(2,"          page %d\\n\\n"
                ,page);
        row=row+3;
        while (row<(FORML-2))
        {
            cpstr=lines[i];
            col=5;
            j=0;
            fprintf(2,"\\r%3d: ",i);

```

(Continued on page 76)

Introducing the Creative Genius...

YOU. Discover how easy programming can be with DataBurst™

A unique runtime screen processor and source program generator, DataBurst™ will decrease your program development time and increase the value of your application programs. The unique DataBurst™ screen editor provides fast, easy screen design. Program independent screen formats reduce both development and maintenance time.

During execution of your program, DataBurst™ controls all user interaction through one assembly language interrupt service routine, requiring as little as 14K of memory. A true full-screen processor, DataBurst™ allows unlimited design complexity, and brings a mainframe advantage to your IBM® PC.

DataBurst™ is available through your local computer retailer or directly from Key Solutions, Inc. To order directly, please send check or money order for \$225* to Key Solutions, Inc., P.O. Box 2297, Santa Clara, CA 95055. Additional language support (BASIC Compiler and C Compiler) is available for \$40* (Please inquire about release dates for other language interfaces).

IBM® is a registered trademark of IBM Corporation.
DataBurst™ is a trademark of Key Solutions, Inc.

*In California add applicable sales tax.

© Copyright 1984 Key Solutions, Inc.

The Design Tool for the Creative Programmer



KEY SOLUTIONS™

Circle no. 52 on reader service card.

C_to_dBASE

The MISSING LINK

C_to_dBASE is a new development tool that allows you to manipulate dBASE data and index files with programs written in the C language. C_to_dBASE provides more than 70 C language functions including:

- Functions to access and modify dBASE data and index files without using dBASE.
- Powerful C language functions for development of dBASE file management programs.
- A menu-driven sample application program that demonstrates the use of C_to_dBASE.
- Full source code in C.
- No royalties.

Whether you are a beginning or professional programmer, C_to_dBASE is a powerful tool for the development of data base applications. Only \$150.00 (includes source code).

For More Information Or
To Order Call:
800-922-0169



COMPUTER INNOVATIONS, INC.

980 Shrewsbury Avenue,
Tinton Falls, NJ 07724

Prices are subject to change without notice.
dBASE is a trademark of Ashton-Tate.

Circle no. 24 on reader service card.

"This is a beautifully documented, incredibly comprehensive set of C Function Libraries."

— Dr. Dobb's Journal



COMPLETE SOURCES

- **PACK 1: Building Blocks I** \$149
250 Functions: DOS, Printer, Video, Async
- **PACK 2: Database** \$399
100 Functions: B-Trees, Variable Records
- **PACK 3: Communications** \$149
135 Functions: Smart-modem™, Xon/Xoff, Modem-7, X-Modem
- **PACK 4: Building Blocks II** \$149
100 Functions: Dates, Text Windows, Pull-down Menus, Data Compression
- **PACK 5: Mathematics I** \$99
35 Functions: Log, Trig, Square Root
- **PACK 6: Utilities I** \$99
Archive, Diff, Replace, Scan, Wipe (Executable Files only)

Lattice™, Microsoft™, DeSmet™,
C1-86™ Compilers on IBM PC/XT/AT™
Small and Large Memory Models.
Credit cards accepted
(\$7.00 handling/Mass. add 5%)



165 Bedford Street
Burlington, Mass. 01803
(617) 273-4711

NOVUM ORGANUM

Circle no. 90 on reader service card.


```

while ((c=cptr[j++]))
{
    if (c=='\t')
    {
        do
        {
            putc(' ',2);
            col++;
        } while ((col%tab)&&
                (col<(FORMW-1)));
    }
    else if ((col>=(FORMW-1))
            && (c != '\n'))
    {
        putc('\n',2);
        break;
    }
    else
    {
        putc(c,2);
        col++;
    }
    row++;
    col=5;
    if (i==tohere)
    {
        col=0;
        break;
    }
    if (++i >= topcnt)
        i=0;
}
while (row++ <= FORML)
    putc('\n',2);
page++;
row=1;
putc('\r',2);
if (col==0)
    break;
}

/*
 * This comments out the pause at
 * the end of each page, reinstate
 * these two lines of code for
 * single sheet paper feeding.
 */

/*
    printf("\rpage %3d:",page);
    printf(" any key to continue > ");
    c=getchar();
*/

}

/*
 * int help()
 *
 * Command help facility. Display
 * a concise list of the browse

```

```

 * commands with syntax and
 * results shown. If new commands
 * are added make sure you add
 * them here also.
 */

help()
{
    printf(CLEAR);
    printf("\n Browse Command");
    printf("\n Information\n");
    printf("\nCommand Results");
    printf("\n-----");
    printf("\n # redisplay");
    printf("\n with line # in center");
    printf("\n # u go up #");
    printf("\n lines and redisplay");
    printf("\n # d go down #");
    printf("\n lines and redisplay");
    printf("\n b redisplay");
    printf("\n beginning of file");
    printf("\n e redisplay");
    printf("\n end of file or buffer");
    printf("\n # t reset tab");
    printf("\n stops to every # spaces");
    printf("\n and redis");
    printf("\n play screen. Tab stops");
    printf("\n default to");
    printf("\n every 4 spaces");
    printf("\n # n go down #");
    printf("\n pages ( default is one )");
    printf("\n # p go up # p");
    printf("\n ages ( default is one )");
    printf("\n q quit and ");
    printf("\n return to operating system");
    printf("\n # # l list file ");
    printf("\n from first # to second # on");
    printf("\n the system ");
    printf("\n list device ( printer )");
    printf("\n # represents any ");
    printf("\n positive integer");
    printf("\n enter any command or");
    printf("\n a return to redisplay\n");
}

/* end of the special functions */

```

End Listing

APROTEK 1000™ EPROM PROGRAMMER



only
\$250.00

TECHNICAL
BREAKTHROUGH
NOW ALLOWS A
PRICE
BREAKTHROUGH

A SIMPLE, INEXPENSIVE SOLUTION TO PROGRAMMING EPROMS

The **APROTEK 1000** can program 5 volt, 25XX series through 2564, 27XX series through 27256 and 68XX devices plus any CMOS versions of the above types. Included with each programmer is a personality module of your choice (others are only \$10.00 ea. when purchased with **APROTEK 1000**). Later, you may require future modules at only \$15.00 ea., postage paid. Available personality modules: PM2716, PM2732, PM2732A, PM2764, PM2764A, PM27128, PM27256, PM2532, PM2564, PM68764 (includes 68766). (Please specify modules by these numbers).

APROTEK 1000 comes complete with a menu driven BASIC driver programmer listing which allows READ, WRITE, COPY, and VERIFY with Checksum. Easily adapted for use with IBM, Apple, Kaypro, and other microcomputers with a RS-232 port. Also included is a menu driven CPM assembly language driver listing with Z-80 (DART) and 8080 (8251) I/O port examples. Interface is a simple 3-wire RS-232C with a female DB-25 connector. A handshake character is sent by the programmer after programming each byte. The interface is switch selectable at the following 6 baud rates: 300, 1.2k, 2.4k, 4.8k, 9.6k and 19.2k baud. Data format for programming is "absolute code", (i.e., it will program exactly what it is sent starting at EPROM address 0). Other standard downloading formats are easily converted to absolute (object) code.

The **APROTEK 1000** is truly universal. It comes standard at 117 VAC 50/60 HZ and may be internally jumpered for 220-240 VAC 50/60 AZ. FCC verification (CLASS B) has been obtained for the **APROTEK 1000**.

APROTEK 1000 is covered by a 1 year parts and labor warranty.

FINALLY — A Simple, Inexpensive Solution To Erasing EPROMS

APROTEK-200™ EPROM ERASER

Simply insert one or two EPROMS and switch ON. In about 10 minutes, you switch OFF and are ready to reprogram.

APROTEK-200™ only \$45.00.

APROTEK-300™ only \$60.00.

This eraser is identical to **APROTEK-200™** but has a built-in timer so that the ultraviolet lamp automatically turns off in 10 minutes, eliminating any risk of overexposure damage to your EPROMS.

APROTEK-300™ only \$60.00.

APROPOS TECHNOLOGY

1071-A Avenida Acaso, Camarillo, CA 93010

CALL OUR TOLL FREE ORDER LINES TODAY:

1-(800) 962-5800 USA or 1-(800) 962-3800 CALIFORNIA

TECHNICAL INFORMATION: 1-(805) 482-3604

Add Shipping Per Item: \$3.00 Cont. U.S. \$6.00 CAN, Mexico, HI, AK, UPS Blue

Circle no. 8 on reader service card.

FORTH

including SOURCE CODE

Developing your own system? Or, just curious about how things work? Either way, **SOURCE CODE** is a must!

KFORTH was developed for use in microprocessor based controllers used by the U.S. Government. It includes CASE statements, a built-in assembler, and CPM file handling. Best of all, you can change it to fit your needs.

SUPER KFORTH was developed for increased speed. It uses **DIRECT THREADED CODE** and is up to 10x faster. Both are written in assembler and can be assembled using **ASM.COM**. Both generate reentrant and ROMABLE code.

(For use with Z80, 8080, 8085 CPM systems)

FILL OUT COUPON TODAY AND MAIL TO:

DJ

KIMRICH COMPUTER DESIGNS, INC.

10404 Patterson Avenue, Richmond, VA 23233 (804) 741-5930

☐ YES! I want **SOURCE CODE!** Enclosed is my check for:

☐ **KFORTH** \$39.95 ☐ **SUPER KFORTH** \$79.95
(In VA add \$1.60 sales tax (4%)) (In VA add \$3.20 sales tax (4%))

My disk format is: (Call for availability of other formats)

☐ 8 inch SSSD ☐ Osborne SD ☐ Kaypro
☐ 5-1/4 inch SSSD ☐ Osborne DD

For VISA or MasterCard orders phone (804)741-5930.

Name _____

Address _____

City _____ State _____ Zip _____

Circle no. 53 on reader service card.

MicroMotion

MasterFORTH

It's here — the next generation of MicroMotion FORTH.

- Meets all provisions, extensions and experimental proposals of the FORTH-83 International Standard.
- Uses the host operating system file structure (APPLE DOS 3.3 & CP/M 2.x).
- Built-in micro-assembler with numeric local labels.
- A full screen editor is provided which includes 16 x 64 format, can push & pop more than one line, user definable controls, upper/lower case keyboard entry, A COPY utility moves screens within & between lines, line stack, redefinable control keys, and search & replace commands.
- Includes all file primitives described in Kernigan and Plauger's *Software Tools*.
- The input and output streams are fully redirectable.
- The editor, assembler and screen copy utilities are provided as relocatable object modules. They are brought into the dictionary on demand and may be released with a single command.
- Many key nucleus commands are vectored. Error handling, number parsing, keyboard translation and so on can be redefined as needed by user programs. They are automatically returned to their previous definitions when the program is forgotten.
- The string-handling package is the finest and most complete available.
- A listing of the nucleus is provided as part of the documentation.
- The language implementation exactly matches the one described in *MASTERING FORTH*, by Anderson & Tracy. This 200 page tutorial and reference manual is included with MasterFORTH.
- Floating Point & HIRES options available.
- Available for APPLE II/II+ /Ile & CP/M 2.x users.
- MasterFORTH — \$100.00. FP & HIRES — \$40.00 each
- Publications
 - *MASTERING FORTH* - \$20.00
 - 83 International Standard — \$15.00
 - FORTH-83 Source Listing 6502,Z-80,8086 - \$20.00 each.



Contact:

MicroMotion
12077 Wilshire Blvd., Ste. 506
Los Angeles, CA 90025
(213) 821-4340

Circle no. 62 on reader service card.

An Introduction to Parsing

by Dr. Henry A. Seymour

What is parsing and why would a programmer want or need to learn about it?

Well, parsing is the process of breaking down an input string into its most elementary parts, referred to as tokens. The portion of a program that performs this action is called the parser. Parsers are used in many areas of computing:

- (1) A compiler translates a high-level language such as Basic or Fortran into object code. Usually one Fortran statement translates into about six object code instructions.
- (2) An assembler translates an assembly language program into object code instructions. The assembly language is machine dependent; usually each instruction is translated into one machine instruction.

program with indentations at the appropriate places.

- (6) A command language processor is a program that accepts the job control language of the operating system and determines the meaning of the request.
- (7) A query language processor is a program that accepts English language requests, determines their meaning, and performs the inquiry from a data base.
- (8) A text editor is a program that accepts a string of commands and, based upon those commands, creates or modifies a file.

All of these applications are interesting enough to discuss in detail; however, because most readers are probably familiar with an assembly language, I will use the assembler as the vehicle of demonstration. Knowledge obtained in the

What smart databases, adventure games, Basic interpreters and Latin teachers have in common.

(3) An interpreter is an operation similar to the assembler, but the computer executes the machine instruction immediately; the compiler and assembler produce object code for later manipulation.

(4) A translator is a program that takes as input a source language and produces an equivalent version in the same language or in a different language; for example, Fortran 66 to Fortran 77, RPG to COBOL, etc.

(5) A pretty printer is a program that takes as input a source program, such as Pascal, and outputs the same pro-

gram with indentations at the appropriate places.

Assembler

In the parsing process, an input string first must be scanned to obtain the tokens of data, then the tokens must be evaluated to determine whether they are meaningful. For example, the input string

LOOP	LOAD	VAL,5
------	------	-------

must be scanned, and the tokens must be isolated:

LOOP	LOAD	VAL	,	5
------	------	-----	---	---

Dr. Henry A. Seymour, Martin Marietta Aerospace, P.O. Box 6184, Huntsville, AL 35806.

Then a decision can be made as to whether the string of tokens represents a valid instruction.

The first step in writing an assembler is to state the characteristics of the assembly language. The language that I will describe does not represent an existing language but is for demonstration purposes only. It is, however, similar to many assembly languages available today. The format is:

[label] operation operand [,register]

The brackets indicate that the enclosed field is optional. Characteristics of the language are:

- Blanks and commas are delimiters.
- The label must begin in column one.
- Each field is either numeric or alphabetic.

The second step is to analyze the language's characteristics to determine its logical structure. A graphic display of this logical structure is called a transition diagram. Figure 1 (below) shows the logical structure of the language described above. The characteristics of a transition diagram are:

- Circles are called states.
- Arrows indicate transition paths.
- Double circles mean a terminating state.

The characters associated with the arrows cause control to move from one state to another. The characters within the circles represent the state type:

- DS is the delimiter state (the beginning state).
- SS is the symbol state.
- NS is the number state.
- TS is the terminal state (ending state).

Let's go through both paths of the transition diagram to see if it will accept the language defined above. We begin at the delimiter state, DS. If the first character is alphabetic, control proceeds to the symbol state, SS, which contains the intelligence of the program. Control at SS implies that an alphabetic field is being parsed.

The looping arrow returning to SS means that control will accept any number of alphabetic characters and will remain in SS. However, upon encountering either a blank or a comma, control will proceed to the terminal state, TS. This means that the process has arrived at a point where a token

has been obtained and can be saved in some location. To allow the next token to be retrieved, control is given back to the delimiter state, DS.

Now consider an alternate path: If a digit is encountered, control will proceed to NS. Being at NS implies that a digit has been found, and control will remain at NS as long as digits are encountered. Upon recognizing a blank or a comma, control will proceed to the terminal state.

The transition diagram will recognize only one unit of information at a time: a number, a symbol, or a delimiter (either the blank or the comma). This is a simplistic model that doesn't consider error conditions. I will discuss that topic later on.

The Pascal-like program in Listing One (page 82) represents the logic in the transition diagram. The functions BLANK, COMMA, ALPHA, and NUMBER test the character to determine its class. The assumption of the program is that the input record is read into an array. The procedure GET_CHAR will move a character from the input array into CHAR for later testing by BLANK, COMMA, ALPHA, and NUMBER. After

the character has been tested it will be moved into the token array by the procedure MOVE_CHAR.

The point to recognize here is that the transition diagram has helped a great deal in describing the program logic. To check for more complex structures in the assembly language and for error conditions, the program must become very large and hence more difficult to read. In that case, the transition diagram will be an even more important aid in the program design process.

Although this approach to the implementation of a parser is preferable to having no developed plan at all, a better and simpler approach gives greater control over the parsing process. That approach involves one additional step: the creation of a state diagram.

State Diagram

The state diagram is equivalent to the transition diagram, but it can be implemented with greater ease and with less source code. The state diagram is a two-dimensional representation of the transition diagram. Each state—that

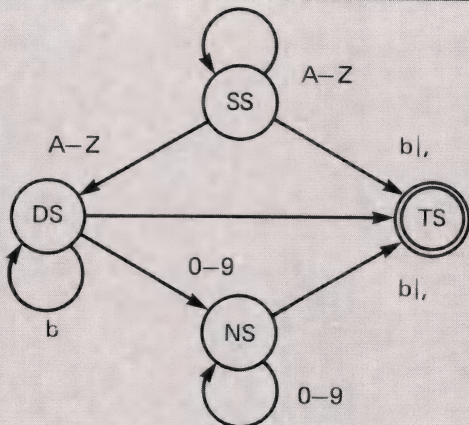


Figure 1.
Transition Diagram

		Input Character			
		b	A-Z	0-9	
State	DS	DS	SS	NS	TS
	SS	TS	SS	TS	TS
	NS	TS	TS	NS	TS

Figure 2.
State Diagram

is, each circle—in the transition diagram is represented as a row in the state diagram. The terminal states—that is, double circles—are the exception and are represented as rows. The transition diagram in Figure 1 would be represented by the state diagram in Figure 2 (page 79).

The contents of the array are the states that may be reached. The terminal state does not need a row because, once it is recognized, there is no reason to continue using the array. In the process of transforming the transition diagram into the state diagram, two error conditions became obvious. When in SS, no exit path exists for a digit, and when in NS, no exit path exists for a symbol. Temporarily, I will place the state code TS in the appropriate cells; later I will discuss how to specify error states.

As shown in Figure 3 (below), the input record is held in an array called an input buffer, and the parsed token is placed into a token buffer. The source record is read into the input buffer, and a pointer is used to point to each position. The program scans the input buffer, copying characters into the token buffer. Upon encountering a de-

limiter, the scanning process stops: the token buffer contains a unit of information.

After the token has been parsed, control is returned to the parsing process with the input pointer pointing to the character that caused the temporary halt. The process begins again at the delimiter state, and another token is parsed. The process continues until an end-of-line condition is detected.

The program in Listing Two (page 82) represents the state diagram shown in Figure 2. The function TYPE in Listing Two determines the category of the input character, as shown in Figure 2, and expresses that in the form of a column value. The program uses two variables, OLD_STATE and NEXT_STATE, to hold the state code of the token that is presently being constructed and the next state that the program is about to enter. The if statement determines how a token has been recognized. The implementation of the state diagram is efficient, easy to understand, and maintainable.

Error Conditions

The trapping of some errors can be implemented in the state diagram; how-

ever, certain errors should be checked only after the token has been obtained. For example, if the language specifications state that a label and operand variable be six characters or less, then the program must check for this. In the process of making a general but efficient model, however, some capabilities such as counting are unavailable.

The solution is to scan the input buffer, moving characters into the token buffer, until a delimiter is encountered. After the token has been obtained, its length can be determined. If the length is in error, an appropriate error message can be displayed. A typical error check for length would be:

```
FOUND : if OLD_STATE = SS and
        LENGTH(TOKEN) > 6 then
        ERROR( 'LENGTH ERROR');
```

Another type of error is the combining of two tokens. In the present specifications, it is invalid to mix letters and numbers, such as LOAD5. The present state diagram will go to TS upon encountering the 5, but, upon reaching the FOUND label, the error will not be obvious. A token of 5 will be found next, and it will be up to another portion of the program to determine whether an error has been found. This kind of error causes the program to assume that the entire field has been obtained, which invalidates the remaining parsing operation. It can also cause multiple error statements to be printed when, in fact, only one error exists.

The state diagram could be rewritten to give it more error checking capability. This would be important if the programmer wanted to make the processor user friendly. How user friendly a program is depends on its ability to recognize errors, identify them to the user, and if possible make corrections. However, the more user friendly the program, the larger and more complex it is.

The state diagram in Figure 4 (at left) includes two new states, E1 and E2. The state diagram now has the ability to trap the two error conditions, such as A1 and 1A. There are no rows for these new states, and they will be treated similarly to the state TS; that is, when the program encounters E1 or E2, it will discontinue the use of the state diagram and proceed to a portion of the program that handles the error

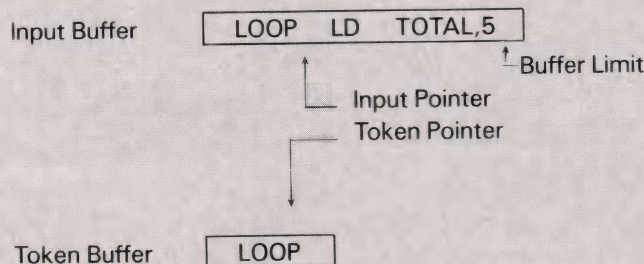


Figure 3.
Data Structures

		Input Character			
		b	A-F	0-9	
State	DS	DS	SS	NS	TS
	SS	TS	SS	E1	TS
	NS	TS	E2	NS	TS

Figure 4.
State Diagram with Error Traps

Dr. Dobb's Journal

A. Your primary job function: (Check one only)

- ☐ Company management (Pres., V.P., Treas., Owner, Gen. Mgr., Mktg. Dir.)
☐ Computer systems management (V.P. EDP, MIS Director, Data Processing Mgr., Data Communications Mgr., Network Planner)
☐ Engineering management (V.P. Engr., Chief Engr., Tech. Director, Dir. R&D)
☐ Systems integrators (Systems Designer, Project Engr., Systems Application Engr., Technical Staff Members)
☐ Consultants (Computer/EDP/Data Communications Consultant)
☐ Educators (Educational users and instructors of computer technology)
☐ Systems/Programming specialists—mini-micro systems
☐ Other (Please specify) _____

B. Which languages are you MOST interested in?

- ☐ BASIC ☐ C ☐ PL/I
☐ Fortran ☐ LISP ☐ APL
☐ COBOL ☐ Prolog ☐ Logo
☐ Pascal ☐ Ada ☐ Smalltalk
☐ Modula-2 ☐ Forth ☐ Other _____

C. What is the operating system?

- ☐ CP/M (or derived)
☐ UNIX (or derived)
☐ MS-DOS (or derived)
☐ Other _____

D. Please indicate which of the following micro-computers you currently own and/or plan to buy in the next 12 months.

	Own	Plan to Buy
Apple	<input type="checkbox"/>	<input type="checkbox"/>
Commodore	<input type="checkbox"/>	<input type="checkbox"/>
Digital Equipment/DEC	<input type="checkbox"/>	<input type="checkbox"/>
Heath/Zenith	<input type="checkbox"/>	<input type="checkbox"/>
Hewlett-Packard	<input type="checkbox"/>	<input type="checkbox"/>
IBM	<input type="checkbox"/>	<input type="checkbox"/>
Macintosh	<input type="checkbox"/>	<input type="checkbox"/>
Radio Shack/Tandy TRS 80	<input type="checkbox"/>	<input type="checkbox"/>
Texas Instruments	<input type="checkbox"/>	<input type="checkbox"/>
Other (Specify)	<input type="checkbox"/>	<input type="checkbox"/>
None	<input type="checkbox"/>	<input type="checkbox"/>

E. What best describes the work you do with this microcomputer?

- ☐ Business functions
☐ Software/Hardware development
☐ Scientific/Engineering/R&D applications

F. Are you the decision maker or very influential in computer-related purchases at work?

- ☐ Yes ☐ No

G. Is your company a dealer, distributor, or systems house for microcomputers?

- ☐ Yes ☐ No

H. Subscriber

- ☐ Yes ☐ No

To obtain information about products or services mentioned in this issue, circle the appropriate number listed below. Use bottom row to vote for best article in issue. One card per person.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	24	25	26	27	
28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54
55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81
82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108
109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135

Articles: 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214

Name _____ Phone (____) _____

Address _____

City/State/Zip _____

Dr. Dobb's Journal

Reader Service Card

A. Your primary job function: (Check one only)

- ☐ Company management (Pres., V.P., Treas., Owner, Gen. Mgr., Mktg. Dir.)
☐ Computer systems management (V.P. EDP, MIS Director, Data Processing Mgr., Data Communications Mgr., Network Planner)
☐ Engineering management (V.P. Engr., Chief Engr., Tech. Director, Dir. R&D)
☐ Systems integrators (Systems Designer, Project Engr., Systems Application Engr., Technical Staff Members)
☐ Consultants (Computer/EDP/Data Communications Consultant)
☐ Educators (Educational users and instructors of computer technology)
☐ Systems/Programming specialists—mini-micro systems
☐ Other (Please specify) _____

B. Which languages are you MOST interested in?

- ☐ BASIC ☐ C ☐ PL/I
☐ Fortran ☐ LISP ☐ APL
☐ COBOL ☐ Prolog ☐ Logo
☐ Pascal ☐ Ada ☐ Smalltalk
☐ Modula-2 ☐ Forth ☐ Other _____

C. What is the operating system?

- ☐ CP/M (or derived)
☐ UNIX (or derived)
☐ MS-DOS (or derived)
☐ Other _____

D. Please indicate which of the following micro-computers you currently own and/or plan to buy in the next 12 months.

	Own	Plan to Buy
Apple	<input type="checkbox"/>	<input type="checkbox"/>
Commodore	<input type="checkbox"/>	<input type="checkbox"/>
Digital Equipment/DEC	<input type="checkbox"/>	<input type="checkbox"/>
Heath/Zenith	<input type="checkbox"/>	<input type="checkbox"/>
Hewlett-Packard	<input type="checkbox"/>	<input type="checkbox"/>
IBM	<input type="checkbox"/>	<input type="checkbox"/>
Macintosh	<input type="checkbox"/>	<input type="checkbox"/>
Radio Shack/Tandy TRS 80	<input type="checkbox"/>	<input type="checkbox"/>
Texas Instruments	<input type="checkbox"/>	<input type="checkbox"/>
Other (Specify)	<input type="checkbox"/>	<input type="checkbox"/>
None	<input type="checkbox"/>	<input type="checkbox"/>

E. What best describes the work you do with this microcomputer?

- ☐ Business functions
☐ Software/Hardware development
☐ Scientific/Engineering/R&D applications

F. Are you the decision maker or very influential in computer-related purchases at work?

- ☐ Yes ☐ No

G. Is your company a dealer, distributor, or systems house for microcomputers?

- ☐ Yes ☐ No

H. Subscriber

- ☐ Yes ☐ No

To obtain information about products or services mentioned in this issue, circle the appropriate number listed below. Use bottom row to vote for best article in issue. One card per person.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	24	25	26	27	
28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54
55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81
82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108
109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135

Articles: 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214

Name _____ Phone (____) _____

Address _____

City/State/Zip _____

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT #27346, PHILADELPHIA, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE TOOLS FOR ADVANCED PROGRAMMERS

Dr. Dobb's Journal

P.O. BOX 13851

PHILADELPHIA, PA 19101

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT #27346, PHILADELPHIA, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE TOOLS FOR ADVANCED PROGRAMMERS

Dr. Dobb's Journal

P.O. BOX 13851

PHILADELPHIA, PA 19101



PUT YOUR PRINTER ON A PEDESTAL.

80-column pedestal \$19.95
132-column pedestal \$24.95
Plus \$1.00 shipping and handling
California residents add 6.5% sales tax

TO ORDER

CALL TOLL-FREE: 800-824-7888

OPERATOR 766

VISA and MASTERCARD ACCEPTED
MONEY BACK GUARANTEE

Zavie
Zavie Enterprises

484 Lakepark Avenue, Suite 184, Oakland, CA 94610

Circle no. 120 on reader service card.

A Software Bridge



DOWNLOADING SERVICE

- Port-A-Soft provides the service of taking programs from a diskette that a customer's computer cannot read and transferring it to a diskette that the customer's computer can read.
- Service available for approximately 250 diskette and tape formats from over 13 micro, mini, and main-frame operating systems.
- Disk to disk, tape to disk, disk to tape conversions.
- Fast service: One day disk conversions, 72-hour tape conversions.
- Competitive prices: Disk conversions as little as \$5.00 per disk plus setup, shipping and handling.

DOWNLOADING SOFTWARE

- Port-A-Soft sells programs that make it possible for the customer's computer to read diskettes for many other computer makes and models.

DOWNLOADING HARDWARE

- Port-A-Soft sells specially designed computers and peripherals that support the reading, writing, and formatting of diskettes for many computer makes and models.

WHO CAN BENEFIT?

USERS: Enhance the power of your micro with programs not available in your diskette format, or with data such as mailing lists, taken from 9 track tapes.

MANUFACTURERS AND DEALERS: Let us help you make that sale that is conditioned on converting the customers data to the new computer, or let us help you provide the sale clinching software that the customer needs to opt for your product.

SOFTWARE PUBLISHERS: Expand your profits by letting us download your software to those unusual formats you cannot afford to support directly, or by porting your software to other operating systems and new markets.

USERS GROUPS: Get the public domain software you want in the format you want for your users group.

PORT-A-SOFT

423 E. 800 N. Orem, Utah 84057 (801) 226-6704

Circle no. 72 on reader service card.

THE LEGENDARY WAY TO SOLVE YOUR BACKUP PROBLEMS

Here's what Microsystems had to say about our original product: "QBAX will probably become one of those legendary programs that everyone eventually buys. It performs a function useful to anyone with a CP/M system, does it well and quickly, is understandable to the novice computer user."

"Every time you run QBAX, the program determines which of your disk files has been changed since the last time it was run. Then it copies these files, and **only** these files, to whatever disk you specify. This is called **incremental backup**, and is the backup method of choice on most large timesharing systems. It will work on any or all active user

©1983 by Ziff-Davis Publishing Company

Amanuensis, Inc.
R.D. #1 Box 236
Grindstone, PA 15442
(412) 785-2806



Qbax TM Amanuensis, Inc.
CP/M Registered TM Digital Research

areas, and so is an absolute **must** for hard- or RAM-disk owners."

Announcing a major enhancement, Qbax2, specifically designed for hard disk users:

- incremental backup by extent
- splits files larger than one floppy
- smart restore: knows exactly which floppies to mount. Can restore individual files or wildcards
- volume space recovery: prevents consuming floppies endlessly when the same files are backed up repeatedly.
- time & date stamp & version number on all backup records.

Qbax2 is \$95. For floppy disk users Qbax1 is still available at \$40.

For CP/M 2.2 on 8" SSSD
& popular 5 1/4" formats
MC, Visa accepted
OEM inquiries invited

Shipping:
\$2 U.S. & Canada, \$4 overseas.

Circle no. 5 on reader service card.

conditions.

The code associated with the FOUND label now can check to determine whether either of these errors has occurred and, if so, what information may be transmitted to the user. The code in Listing Three (page 86) shows one approach to this.

Expanded Assembly Language

Let's increase the strength of the pseudo-assembly language and create a state diagram that will recognize all possible valid tokens. The new instruction format is:

[label] operation operand [,register]

The characteristics of the language follow:

- Blanks and commas are delimiters.
- The operand and register may be a symbol, an integer value, or a hexadecimal value.
- The X followed by a string indicates a hexadecimal value (X'1F').
- The operation field must be a symbol.
- The label must begin in column one.
- The instruction is free form but must be stated completely on one record.
- The operand field may contain one or two operands separated by one of the following arithmetic operators: *, -, +, and /.

- The operand may contain a literal, such as RSTU.

By analyzing these characteristics, one can begin to design the transition diagram. The first noticeable characteristic is that variables are made of letters and numbers. Also, numeric, hexadecimal, and character fields all use the common alphabetic and numeric set of characters. This will be an important feature in the transition diagram.

A study of these features determines the states that must be defined, the paths between the states, and possible error conditions. The transition diagram in Figure 5 (page 83) represents a program that can recognize tokens as well as some error and warning conditions. The error and warning features include a suffix digit, which uniquely identifies the condition that has been detected. Also included is the ability to recognize the end of the input line. The meaning of each state code in Figure 5 is as follows:

- BS is the blank state (beginning state).
- DS is the delimiter state.
- XS is the X state (might be a symbol or hexadecimal state later).
- HS is the hexadecimal state.
- SS is the symbol state.
- NS is the numeric state.
- QS is the quote state.
- OS is the operator state.

In designing the transition diagram, I specified that a hexadecimal string have only the letters A through F and the numbers 0 through 9. Because the transition diagram cannot determine the running value of a hexadecimal number or a decimal number that is being parsed, this type of error trap must be expressed as source code. It is possible to trap the error condition when a hexadecimal string contains an invalid character, such as the letters G through Z. This same error can be trapped at a later point in the program. The programmer may choose where to place the trap. The error states, warning states, and terminal states do not have an equivalent row representation as do the other states. These states must be trapped by the program and appropriate action taken.

Expanded State Diagram

The columns in the expanded state diagram (Figure 6, page 84) are almost as

Procedure GET_TOKEN;

```
DS : GET_CHAR (* from input buffer to CHAR *);
    if BLANK(CHAR) then go to DS
    else if ALPHA(CHAR) then go to SS
    else if NUMBER(CHAR) then go to NS
    else if COMMA(CHAR) then go to TS;

SS : MOVE_CHAR (* from CHAR to token buffer *);
    GET_CHAR (* from input buffer to CHAR *);
    if ALPHA(CHAR) then go to SS
    else if BLANK(CHAR) or COMMA(CHAR) then go to TS;

NS : MOVE_CHAR (* from CHAR to token buffer *);
    GET_CHAR (* from input buffer to CHAR *);
    if NUMBER(CHAR) then to NS
    else if BLANK(CHAR) or COMMA(CHAR) then go to TS;

TS : (* do work with recognized token *)
    ...
```

Listing One

```
OLD_STATE := DS;
repeat
    GET_CHAR (* from input buffer to CHAR*);
    COLUMN := TYPE(CHAR);
    NEXT_STATE := ARRAY(OLD_STATE, COLUMN);
    if NEXT_STATE ≠ DS then
        if TERMINAL(NEXT_STATE) then go to FOUND
        else
            begin
                MOVE_CHAR (* from CHAR to token buffer *);
                OLD_STATE := NEXT_STATE;
            end
        else null (* skip blanks *);
    until INPUT_POINTER > BUFFER_LIMIT;
FOUND :
    (* Token has been found.
    Type of token is described by the contents
    of old_state. *)
```

Listing Two

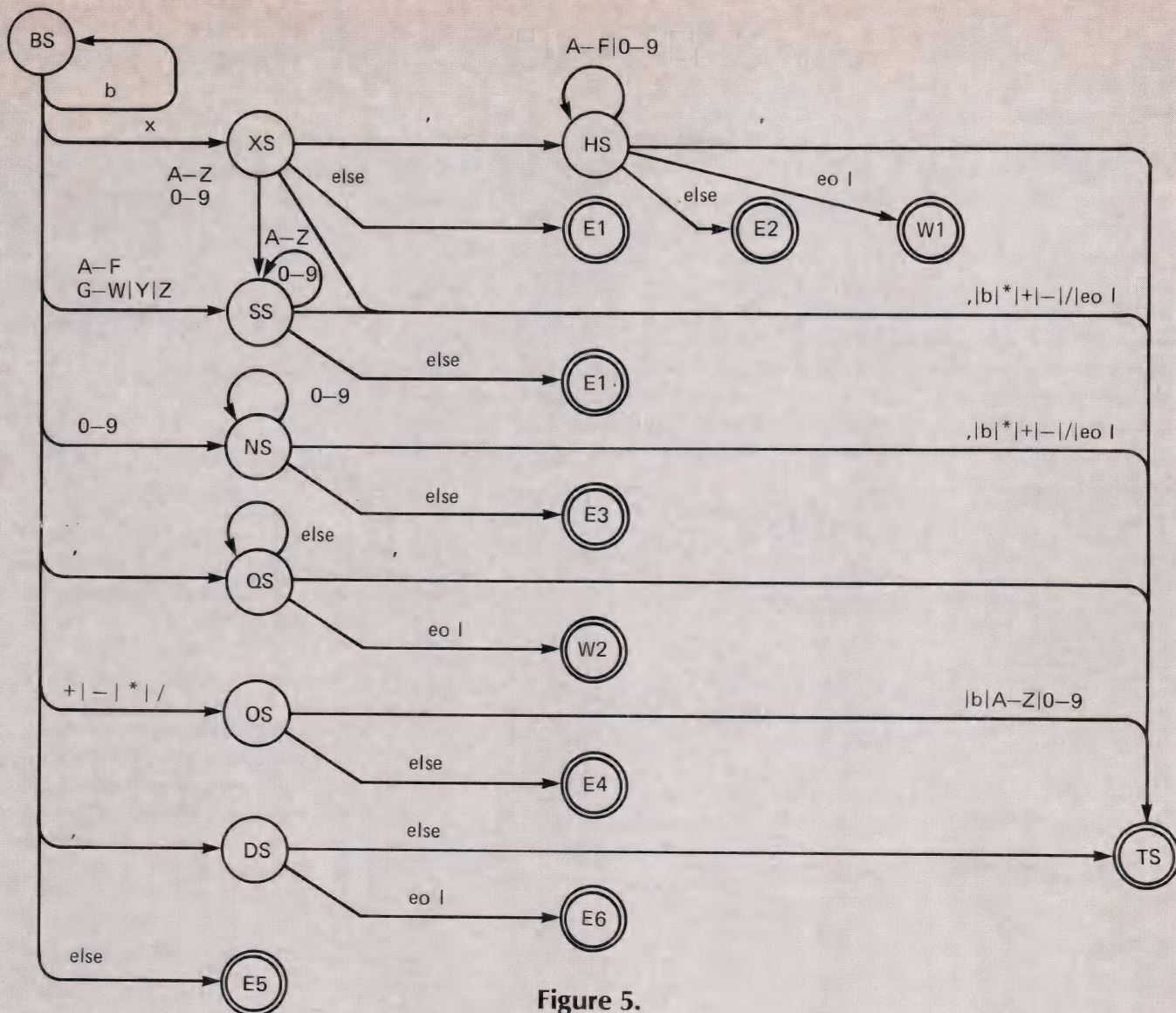


Figure 5.
Transition Diagram

"C/80 . . . the best software buy in America!"

—MICROSYSTEMS

Other technically respected publications like *Byte* and *Dr. Dobb's* have similar praise for **The Software Toolworks' \$49.95** full featured 'C' compiler for CP/M® and HDOS with:

- I/O redirection
- command line expansion
- execution trace and profile
- initializers
- Macro-80 compatibility
- ROMable code
- and much more!

"We bought and evaluated over \$1500 worth of 'C' compilers . . . C/80 is the one we use."

— Dr. Bruce E. Wampler
Aspen Software
author of "Grammatik"

In reviews published worldwide the amazing **\$49.95 C/80** from **The Software Toolworks** has consistently scored at or near the top — even when compared with compilers costing ten times as much!

The optional **C/80 MATHPAK** adds 32-bit floats and longs to the C/80 3.0 compiler. Includes I/O and transcendental function library all for only **\$29.95!**

C/80 is only one of 41 great programs each **under sixty bucks**. Includes: LISP, Ratfor, assemblers and over 30 other CP/M® and MSDOS programs.

For your **free** catalog contact:

The Software Toolworks
15233 Ventura Blvd., Suite 1118,
Sherman Oaks, CA 91403 or call 818/986-4885 today!

CP/M is a registered trademark of Digital Research.

Circle no. 91 on reader service card.

straightforward as before. One must have a column for each unique type of data that is expected, plus a generic "else" column for unexpected or erroneous data. In many cases, each unique edge in the transition diagram in Figure 5 corresponds to a column in the state diagram in Figure 6.

However, as mentioned earlier, the definition of variables, hexadecimal strings, character strings, and numbers uses the same character set. The X must have a column to note the possible beginning of a hexadecimal string. Because X might also be the beginning of a variable, it will be necessary to de-

termine whether the program has arrived at a terminal state and OLD_STATE is XS. This means that the X was recognized, the transition to state X was taken, and a symbol other than a quote caused the transition to the terminal state. In fact, the X is a symbol, and OLD_STATE must be changed to SS. A variable can include any of the alphabet and the digits, but because the digits must be separately recognized they must have their own column. The characters that make the hexadecimal string are a subset of the alphabet plus all the digits. As a result, there is no column just for variables or

hexadecimal strings.

The process of combining the operators into one column requires that the program determine, at a later point, which operator it has found. It would be possible to have a column for each of the operators, but the cost of extra memory to represent them is probably not worth it.

The operand field is restricted to a simple set of cases, such as:

A
A + B
A - B
A * B
A / B

For the assembler to handle more complex arithmetic, we would have to delve into operator precedence parsing. An expression parser would build on this work, taking identified tokens as its input.

Error and Warning Messages

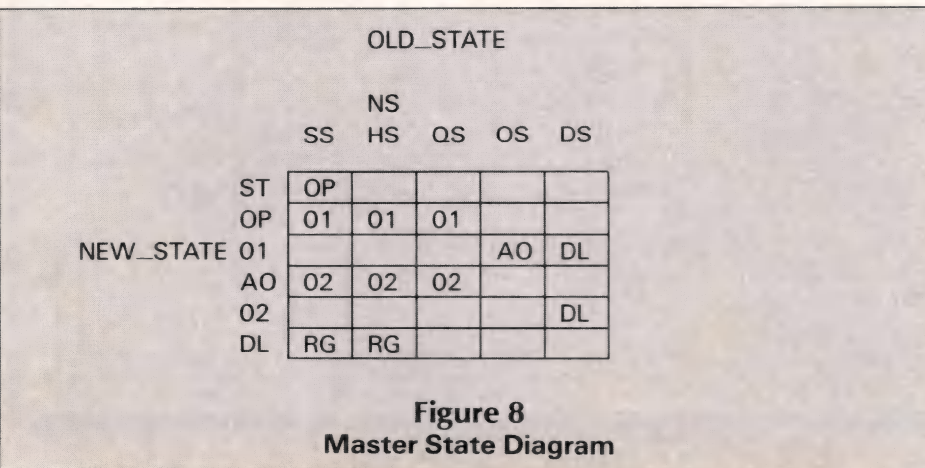
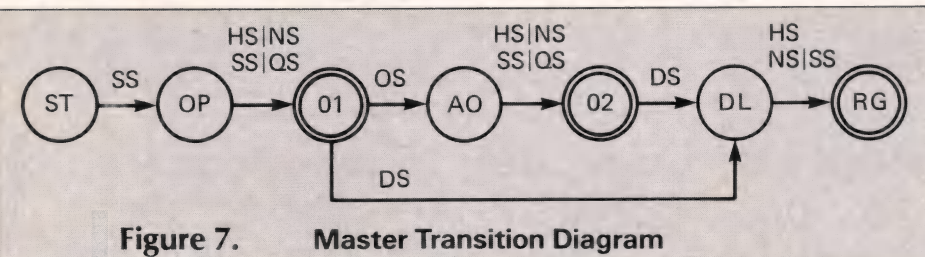
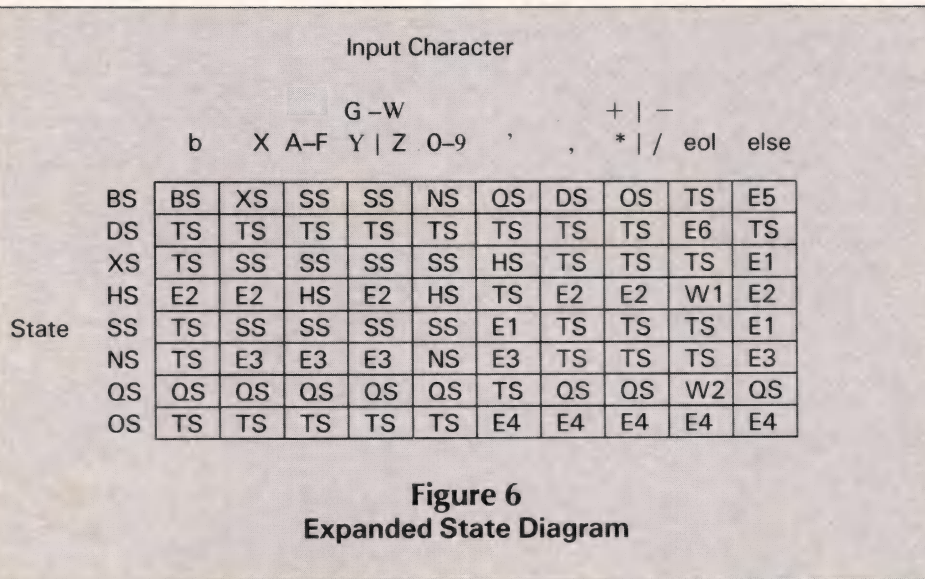
It is possible to have all error traps display one generic statement, i.e., INVALID STATEMENT. Also the program might make no attempt to correct statements, even though it could do so, but this would not be characteristic of a user-friendly program. The error and warning messages that are referred to in Figure 6 are listed as follows:

- E1 - invalid symbol
- E2 - invalid hexadecimal character
- E3 - invalid numeric character
- E4 - invalid arithmetic syntax
- E5 - unrecognizable character
- E6 - missing register
- W1 - missing closing quote in hexadecimal (quote provided)
- W2 - missing closing quote in string field (quote provided)

When a warning state has been reached, the program should repair the string and set the variable OLD_STATE to the appropriate value. For example, if W1 is recognized and the closing quote is provided, then the OLD_STATE should be set to HS.

Master Driver

In the discussion thus far, the main point has been to obtain one token at a time. During the process, error checking determined whether a valid token had been obtained. However, it is pos-



APC MEGABASIC

'The Mercedes-Benz' of BASICs

8086/88 IBM PC AT MS-DOS
CP/M-86 MP/M-86 TURBODOS

NETWORK COMPATIBLE: PCNET, 3-COM, NOVELL, ETC.

MEGABASIC reduces program development time and memory requirements dramatically, executes up to 6 times faster than MBASIC interpreter, is **highly portable among virtually all micro-computers**, and is supported by outstanding documentation.

BENEFITS:

- Addresses up to 1 Mb programs, data.
- Executes as fast as many compilers.
- Superior debugging facilities.
- BCD arithmetic eliminates rounding errors.
- Simple to use—No complicated field statements.
- Source code protection with "scramble" utility.
- Easy-to-use strings (up to 64K long).

THE COMPLETE PACKAGE:

- Developmental version of MEGABASIC in precisions up to 18 digits.
- Run-time semi-compiler version.
- Compaction utility reduces program size.
- Cross-reference generator lists all variables, arrays, subroutines, functions, etc.
- Function library with fast sorts, yes/no prompt routines, matrix manipulation and other routines ready to plug into your programs.
- Configuration program.
- 380-page manual with more than 2,500 index entries.

Complete package: \$400, with 30-day money back guarantee.

**AMERICAN
PLANNING
CORPORATION**

Suite 425
4600 Duke Street
Alexandria, VA 22304
1-800-368-2248
(In Virginia, 1-703-751-2574)

Dealer inquiries invited. VISA or MasterCard accepted.

Circle no. 6 on reader service card.

EVALUATING UNIX*/XENIX** SYSTEMS?

FREE BENCHMARK MANUAL FROM



AIM TECHNOLOGY

California

(800) 672-3470 x: 815

All Other Areas

(800) 538-8157 x: 815

4655 Old Ironsides Drive

Suite 390

Santa Clara, CA 95054

(408) 727-3711

*UNIX is a trademark of Bell Laboratories

**XENIX is a trademark of Microsoft Corporation

Circle no. 4 on reader service card.

UNPARALLELED
PERFORMANCE
and **PORTABILITY**
in an **ISAM PACKAGE**
at an **UNBEATABLE**
PRICE



c-tree™
BY FAIRCOM

2606 Johnson Drive
Columbia MO 65203

The company that introduced micros to B-Trees in 1979 and created ACCESS MANAGER™ for Digital Research, now redefines the market for high performance, B-Tree based file handlers. With c-tree™ you get:

- complete C source code written to K & R standards of portability
- high level, multi-key ISAM routines and low level B-Tree functions
- routines that work with single-user and network systems
- no royalties on application programs

\$395 COMPLETE

Specify format:

8" CP/M® 5¼" PC-DOS 8" RT-II

for VISA, MC or COD orders, call

1-314-445-6833

Access Manager and CP/M are trademarks of Digital Research, Inc.
c-tree and the circular disc logo are trademarks of FairCom

© 1984 FairCom

Circle no. 37 on reader service card.

sible to collect a string of valid tokens without constructing a valid instruction. For example,

5 X'AB' LOAD

is a string of valid tokens, but it is not a valid instruction according to the language specifications.

We can define a transition diagram for valid tokens then convert that to a state diagram. The implementation of the token state diagram to retrieve a token will then perform as a servant to the master state diagram, which deter-

mines the validity of the input instruction. Figure 7 (page 84) shows a transition diagram for a program that determines whether instructions are correct. This master transition diagram makes use of information obtained by the token transition diagram. The transitions from one state to another use the states of the token transition diagram rather than characters. This is a more general view of the input command.

The meaning of each new state symbol in Figure 7 is as follows:

- ST is the start state.
- OP is the operation state.
- O1 is the first operand (terminal

state).

- AO is the operator state.
- O2 is the second operand (terminal state).
- DL is the delimiter state.
- RG is the register state (terminal state).

In Figure 7, at only three states would it be acceptable to terminate: O1, O2, and RG. To stop while at any other state would indicate an incomplete command.

The state diagram shown in Figure 8 (page 84) reflects the logic of the transition diagram in Figure 7. I have left all the error entries blank, assuming that the reader would like to apply the knowledge gained so far by specifying the error codes. The partial program shown in Listing Four (below) is a representation of the state diagram in Figure 8.

The program begins by checking for the presence or absence of the label. If a label is present, it must be entered in a symbol table: hence the need for the special procedure, GET_LABEL. This instruction is not represented in the transition diagram but is implied by the specifications of the language. The program determines at three points whether a valid command has been found. These terminating points are indicated by the double circles in Figure 7 and by the "if input_character = eol" statements in Listing Four.

Summary

The discussion about the general area of parsing has outlined an efficient method of implementing the parsing process:

- Analyze the characteristics of the input data.
- Produce a transition diagram.
- Produce a state diagram.
- Write the source code.

A method of trapping error conditions has been shown, and a coding technique that indicates uniquely which error occurred has been demonstrated. Also discussed was a method of correcting an error condition and giving an appropriate warning message. This information should be applicable to most areas of computing. **DDJ**

```
FOUND :      (* other work here *)
            if NEW_STATE = E1 then
              begin
                ERROR('INVALID SYMBOL');
                (* other work here *)
              end
            else if NEW_STATE = E2 then
              begin
                ERROR('INVALID NUMBER');
                (* other work here *)
              end
            end
```

Listing Three

```
Procedure MASTER_STRING;
...
if column_one_is_not_blank then GET_LABEL;
ST :   GET_TOKEN;
      if OLD_STATE = SS then go to OP
      else ERROR;
OP :   GET_TOKEN;
      if OLD_STATE is in (HS, NS, SS, QS) then go to O1
      else ERROR;
O1 :   GET_TOKEN;
      if input_character = eol then (* found acceptable string *)
      if OLD_STATE = OS then go to AO;
      if OLD_STATE = DS then go to DL
      else ERROR;
AO :   GET_TOKEN;
      if OLD_STATE is in (HS, NS, SS, QS) then go to O2
      else ERROR;
O2 :   GET_TOKEN;
      if input_character = eol then (* found acceptable string *)
      if OLD_STATE = DS then go to DL
      else ERROR;
DL :   GET_TOKEN;
      if OLD_STATE is in (HS, NS, SS) then go to RG
      else ERROR;
RG :   GET_TOKEN;
      if input_character = eol then (* found acceptable string *)
      else ERROR;
```

Listing Four

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 196.



SOFTWARE™

PRESENTS

CP/M®

FOR THE

Macintosh™

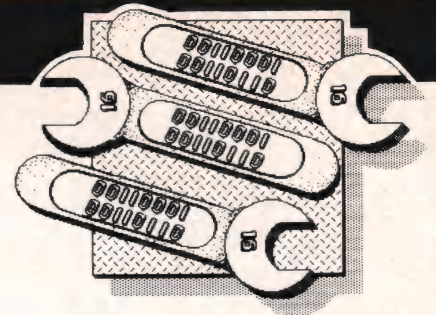
I.Q. SOFTWARE, in one historic move, brings more proven programs to the Macintosh than have existed to date. In the giant world of CP/M based software, **CP/M FOR THE MACINTOSH** delivers the same friendly stand-alone environment to the software developer (professional programmer). If you are wondering why so many developers know and use CP/M it is because CP/M is powerful, easy to learn and easy to use. So easy to use that almost 1 MILLION USERS have CP/M based systems using CP/M based programs and enjoy these same benefits that you will with **CP/M FOR THE MACINTOSH**.



SOFTWARE™

2229 East Loop 820 North
Fort Worth, Texas 76118
(817) 589-2000

CP/M is a registered trademark of Digital Research, Inc.
Macintosh is a registered trademark of Apple Computer, Inc.



by Ray Duncan

Readers Pitch M68000

The 16-Bit Mailbag brought me no less than 10 letters and cards this month from readers requesting more material on the Motorola 68000. Interestingly, not a single one of these readers contributed a 68000 programming tip, listing, or any other words of wisdom. Come on, guys, we aren't operating in a vacuum here!

Those readers who hoped that the introduction of the Macintosh would lead to the development of scads of 68000 public domain software are going to be sadly disappointed. In the first place, Apple has outsmarted itself by making program development on the Macintosh hideously difficult. The native high-level languages available for the Mac are (by IBM PC or even by Z80 CP/M standards) incredibly weak, bug-ridden, slow, and nonstandard. For example, MacBASIC can't even run the *BYTE* Sieve of Eratosthenes benchmark because the Mac runs out of memory. As its latest practical joke (or maybe this is just a nose-thumbing gesture at the free-lance software developer), Apple has released an assembler for the Mac that won't run on just one Mac—you need two. It seems a little incredible, no matter how badly the Mac's 68000 is crippled with overblown operating system software, that the self-proclaimed wizards at Apple couldn't get a two-pass assembler to run on a third-generation microprocessor equipped with 128K of RAM and a 300K+ disk drive. The old Digital Research 8080 assembler ran nicely in 32K with room left over for the operating system.

Let's travel back in time to the February 1984 *BYTE* magazine, in which Steve Jobs was quoted as saying (page 63): "This is an IBM video board; it's only video, nothing else. It's 69 integrated circuits, more chips than an en-

tire Macintosh, and it basically does nothing. And it doesn't even do that very well." Talk about hubris! Remember, that quote was from the same guy who brought you an assembler that requires two computers.

It's becoming clear that the Macintosh's fate will be similar to the fate of the Lisa—critical acclaim, but lackluster sales. My inside source in Cupertino, Deep Golden Delicious, tells me that in the year since the Mac's announcement, approximately 200,000 machines have been delivered. Think back to all that hoopla we were bombarded with last winter, about a super Macfactory for Macs that can grind out one Mac every 15 seconds. Consider that in the same time period, more than a million each IBM PCs and Apple IIs went out the door. Now, I suppose I'm going to get piles of nasty letters from the MacWorshipper crowd. At least then we'll have 68000 topics galore to write about, won't we?

Some 8086 Debugging

Hidden in the obscurities of the Intel 8086 instruction set are some classic booby traps that can take hours (yes, even days) to debug. Here are two to watch out for.

Consider the assembly code in Listing One (page 93). This is the source for a Forth *ROLL* command, which picks a word out of the interior of the machine stack and moves it to the top of the stack. Clue No. 1: As written here, the command executes correctly—most of the time. Clue No. 2: In the Intel iAPX 86,88 *User's Manual* (page 2-42), the fine print says that "execution does not resume properly [after an interrupt] if a second or third prefix . . . has been specified in addition to any of the repeat prefixes." Aha! This code fails because, at unpredictable intervals, a hardware inter-

rupt occurs during the repeated execution of the string instruction. The 8086 loses track of either the addressing context or the repeat prefix itself upon return from the interrupt (depending on the order in which prefixes were assembled); consequently, either the wrong number of words is moved, or the words are moved from the wrong source address.

Now look at Listing Two (page 93). This code is supposed to transfer a number from the top of the 8086's machine stack to the top of the 8087's machine stack. It works correctly "most of the time." The bug here results from a subtle failure of synchronization. Since the programmer failed to include a *WAIT* after the *FLD[BX]* instruction before incrementing the 8086's stack pointer, a hardware interrupt could occur and be serviced after the *ADD SP,8* instruction is executed but before the 8087 has completed its transfer of the number from shared memory. Thus, the process of servicing the interrupt will push the CPU flags and return address on top of the number that the 8087 is loading, partially or completely destroying that number. This type of problem is extremely tough to isolate. Users can best avoid the problem altogether by paying scrupulous attention to synchronization and stack protection.

Sneak 80286 Preview

Since IBM has put its Good Computing Seal of Approval on the Intel 80286 with the introduction of the PC/AT, it behooves us all to start learning to use this processor properly. The 80286, when running in "Protected Virtual Address" mode, is a fearsome beast. It has several new addressing considerations, hardware-recognized data structures and descriptors, and memory protection mechanisms; it is to an 8086 what a VAX is to an LSI-11.

However, the 80286 in "Real Address" mode can be viewed as a slightly tuned up 8086; this is helpful to us aging, simple-minded software developers. Fortunately or unfortunately, PC-DOS and MSDOS 3.0 use the 80286 in Real Address mode, so we can safely ignore the more complex considerations for the present.

Changed Instructions

To start with, let's look at some subtle differences between the instruction sets of the 8086 and the 80286.

- The instruction PUSH SP pushes the current pointer, rather than the new stack pointer. In other words, the 8086 did something like this:

```
PUSH SP =
    SP := SP - 2
    (SP) := SP
```

while the 80286 does something like

```
PUSH SP =
    TEMP := SP
    SP := SP - 2
    (SP) := TEMP
```

So the 80286 instruction

```
PUSH SP
```

has the effect of the 8086 sequence

```
MOV AX,SP
PUSH AX
```

- The 80286 divide error exception (interrupt 0) pushes CS:IP of the instruction that caused the exception. The 8086 pushed the CS:IP of the instruction following the instruction that caused the exception.

- Shift counts are masked to 5 bits. For example, if you put the value 40 in CX and execute

```
SHL AX,CX
```

the contents of AX will be shifted left 8 bits. On the 8086, the processor attempted a left shift of 40 bit positions, and the result in AX would always be zero.

Errant Instructions

There are also some known bugs in the 80286 revision B chips.

- After execution of POPF, a pending maskable interrupt may be improperly recognized, even though maskable interrupts were disabled prior to execution of POPF and the flags word popped from the stack has IF = 0. If the interrupt is improperly recognized, it will, however, be properly executed. This problem is particularly relevant for CP/M-86 system users, since many

implementations of this operating system run without interrupts; if an interrupt is unexpectedly serviced, the interrupt vectors may not have been initialized, and the system will crash.

Apparently, this problem occurs only when the 80286 is running with zero or one wait states. It can be avoided altogether by running the 80286 with two or more wait states (this, however, incurs a significant performance penalty). Alternatively, you can redefine POPF in a way that simulates its action without actually executing the POPF opcode. There are a couple of similar ways to do this; here is the one

given by Intel in its errata sheet:

CodeMacro POPF	;assume flags on stack
PUSH CS	
CALL \$ + 3	;push IP
PUSH BP	;Save BP and
MOV BP,SP	;address the stack
	;add to IP value on stack
	;to point past IRET
ADD WORD PTR [BP + 2],9	
POP BP	;restore BP
IRET	;pop flags, CS, and IP

MEMO: C Programmers

QUIT WORKING SO HARD.

These people have quit working so hard: IBM, Honeywell, Control Data, GE, Lotus, Hospitals, Universities & Government Aerospace.

THE GREENLEAF FUNCTIONS™

THE library of C FUNCTIONS that probably has just what you need . . . TODAY!

- ... already has what you're working to re-invent
- ... already has over 200 functions for the IBM PC, XT, AT, and compatibles
- ... already complete ... already tested ... on the shelf
- ... already has demo programs and source code
- ... already compatible with all popular compilers
- ... already supports all memory models, DOS 1.1, 2.0, 2.1
- ... already optimized (parts in assembler) for speed and density
- ... already in use by thousands of customers worldwide
- ... already available from stock (your dealer probably has it)
- ... It's called the **GREENLEAF FUNCTIONS**.

Sorry you didn't know this sooner! Just order a copy and then take a break — we did the hard work. Already.

THE GREENLEAF FUNCTIONS GENERAL LIBRARY: Over 200 functions in C and assembler. Strength in DOS, video, string, printer, async, and system interface. All DOS 1 and 2 functions are in assembler for speed. All video capabilities of PC supported. All printer functions. 65 string functions. Extensive time and date. Directory searches. Polled mode async. (If you want interrupt driven, ask us about the **Greenleaf Comm Library**.) Function key support. Diagnostics. Rainbow Color Text series. Much, much more. **The Greenleaf Functions.** Simply the finest C library (and the most extensive). All ready for you. From Greenleaf Software.

... **Specify compiler** when ordering. Add \$7.00 each for UPS second-day air. MasterCard, VISA, check, or P.O.



- ◆ Compilers:
 - CI C86 \$349
 - Lattice \$395
 - Mark Williams ... \$475
- ◆ General Libraries.... \$175
 - (Lattice, Microsoft, Mark Williams, CI C86)
- ◆ DeSmet C \$150
- ◆ Comm Library \$160

GREENLEAF SOFTWARE, INC.

2101 HICKORY DRIVE • CARROLLTON, TX 75006 • (214) 446-8641

EndM

- The LOCK prefix is ignored in instructions that reference memory only once (e.g., MOV reg,mem) but works properly for instructions that both read and write memory (e.g., ADD mem,reg or XCHG mem,reg). Thus, when programming the 80286, you should use XCHG to manipulate semaphores.
- The 80286 may fail to generate a protection exception in cases where the beginning of a multibyte operand for the 80287, addressed via DS or ES, lies within an unprotected area but crosses into a protected area. Note that this is

only relevant when the system is running in Protected Virtual Address mode.

All three of these problems have reportedly been fixed in 80286 revision C parts. Revision B-2 chips can be recognized by the copyright marking of "© Intel '83."

Added Instructions

The 80286 microprocessor has several new opcodes. Most of them will be useful only to the authors of compilers or device drivers. A few of them, however, will be helpful to us average Joes, too.

Push Immediate Value

On the 80286, you can code
PUSH 4
which is equivalent to the 8086
MOV AX,4
PUSH AX

Push All (PUSHA)

This 80286 opcode will push all general registers; it is equivalent to the 8086 code:

PUSH AX
PUSH CX
PUSH DX
PUSH BX
MOV AX,SP
PUSH AX
PUSH BP
PUSH SI
PUSH DI

Pop All (POPA)

This 80286 opcode will pop all general registers from the stack; it is equivalent to the 8086 code:

POP DI
POP SI
POP BP
ADD SP,2
POP BX
POP DX
POP CX
POP AX

Note that the contents of register SP that were pushed by the PUSHA instruction are discarded rather than being loaded into SP; this, of course, is vital in saving the stack context.

Signed Multiply by Immediate Value

This was one of the more glaring deficiencies on the 8086; it is remedied on the 80286. For example, you can write

IMUL 10

where on the 8086 you would have had to code something like

MOV BX,10
IMUL BX

Shift/Rotate Memory or Register by Count

For example, on the 80286, you can code

ROL AX,3

where on the 8086 you would have to write either

MOV CX,3
ROL AX,CX

or the sequence

ROL AX,1



dBASE II outfoxed!

Who says the most popular database management system is the best?

Introducing **FoxBASE II™**, the new relational database management system that's dBASE II source compatible. It does everything dBASE II does ... plus a whole lot more.

- Runs 3 to 5 times faster
- Sorts up to 20 times faster
- Permits up to 48 fields per record ... 50% more than dBASE II
- Supports full type-ahead
- Compiles program sources into compact object code
- Comes with a sophisticated online manual and HELP facility
- Has twice as many memory variables

What's more, it costs less. **MS-DOS \$395. AOS/VS \$995.**

FoxBASE II is available now for: IBM-PC, IBM-PC/XT, COMPAQ & IBM compatibles, TI Professional, DG Desktop, DG MV Series, and many others. Call or write today for more information.

Developed by

DACOR

COMPUTER SYSTEMS 13330 Bishop Road, P.O. Box 269, Bowling Green, OH 43402 / 419-354-3981 / TWX 810-499-2989

dBASE II is a registered trademark of Ashton-Tate
FoxBASE II is a trademark of Fox Software Inc.

FoxBASE II
from FOX SOFTWARE INC.

Circle no. 40 on reader service card.

ConIX™

UNIX™ Technology for CP/M™

ConIX can provide any 48K+ CP/M-80 system with many advanced capabilities of UNIX. You'll be amazed at what your CP/M micro can do now! ConIX features include:

I/O Redirection and Pipes (uses memory or disk), multiple commands per line, full upper/lower case and argument processing, Auto Screen Paging, Programmable Function Keys, improved User Area Directory manipulation, Command and Extension (Overlay) Path Searching, "Virtual" disk system, 8Mb Print Spooler, extensive preprocessed "Shell" command programming language, 300+ variables, over 100 built-in commands, Math Package, 22 new BDOS SysCalls, Archiver (compacts files for disk space savings of over 50%), On-Line Manual System, and much more! Uses as little as 1/2K RAM! Runs with CP/M for true data and software compatibility. Installs quickly and easily without any system modifications.

The ConIX Operating System List Price: \$165

Price includes Instructional Manual, 8" SSD disk and free support. Format conversion available. To order, contact your local dealer, or buy direct and add shipping: \$4.50 UPS, \$10 Canada, \$25 overseas, COD \$2 extra (USA only). NY State residents add sales tax.



Computer Helper Industries Inc.
P.O. Box 680 Parkchester Station, NY 10462
Tel. (212) 652-1786

Dealer inquiries invited!

UNIX: AT&T Bell Labs, CP/M: Digital Research, ConIX: Comp. Helper Ind. Inc.

Circle no. 22 on reader service card.

Hello,
I'm Bill Salkin

As editor of PC FIRING LINE/PC UNDERGROUND, the bi-monthly technical disk magazine for the IBM PC, I invite you to join us as we discuss critical



error-handling routines, bicubic splines, drool over "printf" source code, share FREEWARE, provide programmer-oriented DEMOS and tips, dissect utilities like the DOS 2.x EXEC function, continue our ADA, ASM, C, FORTH, and LISP tutorials, and more! Sizzling ready-to-use source code included in each jam-packed issue (currently 2 DS/DD disks per issue!).

- Price: \$12 per issue, or \$72 for a one-year (six-issue) subscription. Make checks payable to ABComputing. (All currency in U.S. dollars. Foreign countries send money orders and add \$5 airmail for each issue.)

- Make non-profit copies for your friends. Those receiving these copies are asked to pay us \$6 to help defray our production costs. Make checks payable to ABComputing.

- Requires 128K RAM, any release of DOS, and one double-sided disk drive. Note: We ship ONLY DS/DD disks.

ABComputing

Dept. 50, P.O. Box 5503, North Hollywood, CA 91616-5503
(818) 509-9002

Circle no. 1 on reader service card.

Six Times Faster!

Super Fast Z80 Assembly Language Development Package

Z80ASM

SLRINK

- Complete Zilog Mnemonic set
- Full Macro facility
- Plain English error messages
- One or two pass operation
- Over 6000 lines/minute
- Supports nested INCLUDE files
- Allows external bytes, words, and expressions (EXT1 * EXT2)
- Labels significant to 16 characters even on externals (SLR Format Only)
- Integral cross-reference
- Upper/lower case optionally significant
- Conditional assembly
- Assemble code for execution at another address (PHASE & DEPHASE)
- Generates COM, HEX, or REL files
- COM files may start at other than 100H
- REL files may be in Microsoft format or SLR format
- Separate PROG, DATA & COMMON address spaces
- Accepts symbol definitions from the console
- Flexible listing facility includes TIME and DATE in listing (CP/M Plus Only)

- Links any combination of SLR format and Microsoft format REL files
- One or two pass operation allows output files up to 64K
- Generates HEX or COM files
- User may specify PROG, DATA, and COMMON loading addresses

- COM may start at other than 100H
- HEX files do not fill empty address space.
- Generate inter-module cross-reference and load map
- Save symbol table to disk in REL format for use in overlay generation
- Declare entry points from console
- The FASTEST Microsoft Compatible Linker available

SPEED!
LINKER!
SPEED!

- Complete Package Includes: Z80ASM, SLRINK, SLRIB - Librarian and Manual for just \$199.99. Manual only, \$30.
- Most formats available for Z80 CP/M, CDOS, & TURBODOS
- Terms: add \$3 shipping US, others \$7. PA add 6% sales tax

For more information or to order, call:

1-800-833-3061

In PA, (412) 282-0864

Or write: SLR SYSTEMS

1622 North Main Street, Butler, Pennsylvania 16001

SLR Systems

Circle no. 82 on reader service card.

ROL AX,1
ROL AX,1

Input and Output String (INS and OUTS)

These are new members of the string instruction group that also includes MOVSB, CMPSB, SCASB, LOCSB, and STOSB. INS transfers data from the port number in the DX register to the memory address represented in ES:DI, while OUTS transfers data from the memory address represented in DS:SI to the port number that is in DX. Both INS and OUTS can transfer either byte or word values and can accept a REP prefix that is controlled by the contents of CX and causes autoincrement or autodecrement of the appropriate index register, depending on the state of the direction flag.

Enter Procedure (ENTER)

This creates a stack frame and initializes a frame pointer. It was added to support the compilation of procedures in block-structured, high-level languages such as PL/I, Pascal and (God forbid) Ada.

Leave Procedure (LEAVE)

This releases a stack frame and restores the previous contents of the frame pointer. It reverses the effect of ENTER.

Detect Value out of Range (BOUND)

This tests whether an array index falls within the range defined by the contents of a two-word block of memory; if not, an interrupt 5 occurs.

There are also 16 new instructions concerned with task concurrency and memory protection that are beyond the scope of this column. They load or store global, local, or interrupt descriptor registers, control write access to regions of memory, and change task privilege levels. There is also a raft of new ways you can use familiar instructions (such as IRET) to generate protection exceptions. These we'll leave for a later, more profound, column.

New Interrupts

The 80286 adds nine new hardwired interrupts to those defined on the 8086. These are:

Interrupt Cause

5 BOUNDS executed with ar-

ray index out of range

- 6 Execution of undefined opcode
- 7 Coprocessor protection error, relevant in Protected Virtual Address mode
- 8 Interrupt table limit fault, or the dreaded Double Fault (e.g., protection fault followed by segment not present fault, such as might be caused if the protection fault interrupt handler had been paged out by the virtual memory manager)
- 9 In Real Address mode, coprocessor data transfer wrap-around past offset 0FFFFH. In Protected mode, this exception will also occur if the first part of a multi-byte 80287 operand falls within an unprotected area but crosses into a protected memory area
- 10 Invalid task state segment; attempted to switch context to a task with an illegal descriptor, Protected mode only
- 11 Memory segment not present—support for virtual memory manager, Protected mode only
- 12 Stack fault—stack overflow or underflow, or stack reference to a memory segment not present, Protected mode only
- 13 In Real Address mode, segment wraparound attempted by a word operation at offset 0FFFFH, or a stack push with SP = 1 during PUSH, CALL, or INT. In Protected mode, general protection fault; any memory protection exception not covered by the other error interrupts.

Faster Microcode

Many of the 80286's instructions that are functionally identical to the 8086 actually execute much faster due to improved implementation. For example, a 16×16 -bit register-register signed multiply, which requires 128–154 clocks on the 8086, requires 21 clocks on the 80286. This concludes our Sneak Preview of the Intel 80286 CPU. More

to come in subsequent columns.

Determining PC Type

As the IBM PC family proliferates, software developers will need a way to determine the type of host machine at runtime. IBM has declared that the ROM location F000:FFFE may be inspected by software and that its content has the following meaning:

Contents	Machine
0FF	IBM PC
0FE	PC/XT
0FD	PCjr
0FC	PC/AT

It would be helpful to know what this location contains on other IBM PC-like models such as the 3270PC and the vast family of IBM-compatibles. My Compaq (an early model, ROM copyright 1982) has 02DH at this location.

If I Had a Hammer

Russ Hayden of Natick, Massachusetts, writes: "... in the June 1984 installment of the 16-Bit Software Toolbox, there are some 8086 assembly routines to convert binary values to ASCII hexadecimal. Ray, if these tools were screwdrivers, they would be made of tinfoil. It's not that they don't work (they'll execute fine), but rather the use of the 'DIV' (divide) instruction to divide a binary number by 16, where four right shifts will accomplish the same purpose. The DIV instruction takes 90 clocks in 8086; four right shifts take a total of eight clocks.

"I don't think we've reached the point in processing power where such things no longer matter, especially in routines likely to be incorporated into larger programs and used frequently. A column on tools should be sensitive to the many ways to approach a problem and their relative merits. [I've enclosed] a suggested improvement in the byte_to_hex routine."

It's really embarrassing to be caught out on this. I've always been a fervent advocate of using shifts and avoiding hardware divides whenever it's remotely feasible. Oh well, see Listing Three (page 93).

DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 197.

FORTRAN PROGRAMMERS

Lahey Computer Systems is pleased to announce

F77L

the complete implementation of the ANSI FORTRAN 77 standard for the IBM PC and compatibles.

With fast compile and execution speeds, specific diagnostics at compile and runtime, F77L meets the needs of mainframe programmers who recognize FORTRAN as the workhorse of computer languages. In addition to meeting the '77 Standard, F77L features include:

- Many IBM H features: \$ in a name, 8 character names. Types: LOGICAL★1, REAL★8, INTEGER★2, COMPLEX★16.
- INCLUDE, OPTION, and CHAIN statements.
- Optional checking: subscript, subprogram class, argument and alternate return count.
- Runtime messages include text and subprogram/line-number traceback.
- COMMONS and subprogram units may be as large as 64K.
- Source file is free format: comments begin with asterisk, continuation lines begin with ampersand.
- Complete and easy to follow 250 page manual.
- Telephone user support and newsletter with updates.

When you purchase the F77L you are buying more than a language system you are also buying LCS's commitment to FORTRAN programming. We have specialized in FORTRAN since 1969 and were the first to implement FORTRAN 77. Because we sell only one product, our customers know that our total effort is directed towards the development of F77L and the continuing service of our users.

If you are serious about FORTRAN programming, then you owe it to yourself to compare Lahey Computer Systems' F77L to the competition.

Complete package: \$477 Visa/MC

To order or for more information call or write:



Lahey Computer Systems, Inc.

904 Silver Spur Road, Suite 417

Rolling Hills Estates, CA 90274 (213) 541-1200

Circle no. 56 on reader service card.

WRITE

The Writer's Really Incredible Text Editor lives up to its name! It's designed for creative and report writing and carefully protects your text. Includes many features missing from WordStar, such as sorted directory listings, fast scrolling, and trial printing to the screen. All editing commands are single-letter and easily changed. Detailed manual included. Dealer inquiries invited. WRITE is \$239.00.

BDS's C Compiler

This is the compiler you need for learning the C language and for writing utilities and programs of all sizes and complexities. We offer version 1.5a, which comes with a symbolic debugger and example programs. Our price is (postpaid) \$130.00.

Tandon Spare Parts Kits

One door latch included, only \$32.50.
With two door latches \$37.50.
Door latches sold separately for \$7.00.

All US orders are postpaid. We ship from stock on many formats, including: 8", Apple, Osborne, KayPro, Otrona, Epson, Morrow, Lobo, Zenith, Xerox. Please request our new catalog. We welcome COD orders.

Workman & Associates

112 Marion Avenue
Pasadena, CA 91106
(818) 796-4401



Circle no. 118 on reader service card.

At Last! bds C ... Ver. 1.5

**Including a new dynamic debugger
Still the choice of professionals**

- Compiler option to generate special symbol table for new dynamic debugger by David Kirkland. (With the debugger, the distribution package now requires two disks.)
- Takes full advantage of CP/M® 2.x, including random-record read, seek relative to file end, user number prefixes, and better error reporting.

V 1.5 \$120.00

V 1.46 \$115.00

(needs only 1.4 CP/M)

Other C compilers and
C related products
available . . . Call!

TERMS: CHECK,
MONEY ORDER, C.O.D.,
CHARGE CARD

HOURS: 9 am—5 pm

Monday—Friday

(316) 431-0018

- Link option to suppress warm-boot
- New library file search capabilities
- New, fully-indexed 180 page manual
- * CP/M is a trademark of Digital Research, Inc.

IT'S HERE!

MONEY MATH

- Uses BCD internal representation.
- You choose from two types of rounding.
- Configurable exception handling
- Distributed with 12 digits precision. Easily configured for more or less
- Excess 64 exponents

SOURCE INCLUDED \$5000



include \$2.50 for postage and handling

Circle no. 32 on reader service card.

A Professional Quality Z80/8080 Disassembler

REVAS Version 3

Uses either ZILOG or 8080 mnemonics
Includes UNDOCUMENTED Z80 opcodes
Handles both BYTE (DB) & WORD (DW) data
Disassembles object code up to 64k long!
Lets you insert COMMENTS in the disassembly!

A powerful command set gives you:

INTERACTIVE disassembly
Command Strings & Macros
On-line Help
Calculations in ANY Number Base!
Flexible file and I/O control
All the functions of REVAS V2.5

REVAS:

Is fully supported with low cost user updates
Runs in a Z80 CPU under CP/M*
Is normally supplied on SSD 8" diskette
Revas V 3...\$90.00 Manual only...\$15.00
California Residents add 6 1/2% sales tax

REVASCO

**6032 Charlton Ave., Los Angeles, CA. 90056
(213) 649-3575**

*CP/M is a Trademark of Digital Research, Inc.

Circle no. 80 on reader service card.

EC THE PROGRAMMER'S TEXT EDITOR

Compile and Edit Your Programs Inside EC!

Now you can edit, compile and test your program from inside the EC editor.* In fact, EC gives you complete access to the operating system. Do a directory, copy or delete files, even run other programs without ever leaving EC!

MULTIPLE WINDOWS—

Forget about dumping a file you're editing just so you can see what is in another file ... open a new window, up to five of them, and read in the file you want to use.

All windows can be shown on the screen at the same time; or the screen can be dedicated to just a single file, while the others are kept in the background—only a keystroke away. You can even cut and paste blocks of text between windows!

FULL SCREEN EDITOR FOR THE IBM—

Developed specifically for the IBM PC, EC makes extensive use of the entire PC keyboard so editing is fast and intuitive.

In addition to standard editing features, EC supports command and text macros, word wrap, paragraph reformatting, horizontal scrolling, and control for color monitors.

DEMO DISK IS ONLY \$5—

Call or write for our full-featured demo. You'll get a standard version of EC that handles up to 10K worth of files and a complete set of documentation.

See for yourself how pleasant it is to use an editor that gives you both unrestricted access to the operating system and multiple windows.

You won't be disappointed!

EC EDITOR - \$125
EC DEMO - \$5 (plus \$1.65 for COD)

* EC runs on an IBM PC, or look-alike, with at least 128K RAM under DOS 1.1 or higher. To use the DOS interface feature you must have DOS 2.0 or higher and enough RAM to run the additional program.

IBM is a trademark of International Business Machines
MO. RESIDENTS ADD 6% SALES TAX



C SOURCE
12801 Frost Road • Kansas City, MO 64138
816-353-8808

16-Bit (Text begins on page 88)

Listing One

What's wrong with this picture?

```
roll proc near
;extract word n from the
;depths of the parameter
;stack, pushing it on top
;of the stack.
pop bx
;get return address out
;of the way.
mov ax,ss
mov es,ax
pop di
;can't override ES, so
;make it address stack segment.
;get number of stack cell
;to bring to the top.
mov cx,di
inc cx
sal di,1
add di,sp
mov si,di
sub si,2
push ss:[di]
;calculate number of stack
;words to slide.
;calc destination address
;for slide.
;calculate source address
;for slide.
;copy the desired cell to
;top of stack.
std
;set direction flag for
;string move.
;now slide the stack.
rep movs es:word ptr [di],ss:word ptr [si]
add sp,2
jmp bx
;return to caller.
roll endp
```

End Listing One

Listing Two

What's wrong with this picture?

```
.
.
.
mov bx,sp
wait
fld ss:[bx]
add sp,8
.
.
.
```

End Listing Two

Listing Three

Improved conversion routine (see also column in June DDJ).

```
byte_to_hex proc near
;convert binary value to
;hex ASCII
;AL=binary value
;DI=pointer to storage
; for string
;save lower nibble
;divide upper nibble by 16

mov ah,al
shr al,1
shr al,1
shr al,1
shr al,1
call ascii
stosb
mov al,ah
and al,0fh
call ascii
stosb
ret

;convert it to ASCII
;and store it
;get back lower nibble
;mask to four bits
;convert and store

byte_to_hex endp
```

End Listings

QUALITY SOFTWARE AT REASONABLE PRICES

CP/M Software by
Poor Person Software

Poor Person's Spooler **\$49.95**

All the function of a hardware print buffer at a fraction of the cost. Keyboard control. Spools and prints simultaneously.

Poor Person's Spread Sheet **\$29.95**

Flexible screen formats and BASIC-like language. Pre-programmed applications include Real Estate Evaluation.

Poor Person's Spelling Checker **\$29.95**

Simple and fast! 33,000 word dictionary. Checks any CP/M text file.

aMAZEing Game **\$29.95**

Arcade action for CP/M! Evade goblins and collect treasure.

Crossword Game **\$39.95**

Teach spelling and build vocabulary. Fun and challenging.

Mailing Label Printer **\$29.95**

Select and print labels in many formats.

Window System **\$29.95**

Application control of independent virtual screens.

All products require 56k CP/M 2.2 and are available on 8" IBM and 5" Northstar formats, other 5" formats add \$5 handling charge. California residents include sales tax.

Poor Person Software

3721 Starr King Circle
Palo Alto, CA 94306
tel 415-493-3735

CP/M is a registered trademark of Digital Research

GGM — FORTH™ has HELP* for Z80¹ using CP/M²

GGM—FORTH, a complete software system for real-time measurement and control, runs on any Z80 computer under CP/M using an extended FORTH vocabulary.

GGM—FORTH features:

- Open multiple CP/M files, in any combination of direct-access and sequential-access, fully compatible with all CP/M utilities
- Char. in/out uses CP/M console, lister, file, or port
- On-line HELP* provides instant access to definitions in the run-time GGM—FORTH dictionary
- HELP* file is easily extended to include user definitions using HELP* utility
- HELP* is available during full-screen editing

Complete system and manuals **\$150.**

Manuals only: **\$ 20.**

Introductory System: **\$ 35.**

GGM SYSTEMS, INC.
135 Summer Ave.,

(617) 662-0550
Reading, MA 01867

¹Z80 is a trademark of Zilog, Inc.

²CP/M is a trademark of Digital Research, Inc.

Circle no. 71 on reader service card.

Circle no. 41 on reader service card.

Super Fast

Get Fast Relief!

S-100! IBM PC/XT! TRS*80 II! EPSON QX10! ZENITH Z-100!

If you've been "patient" with slow disk drives for too long, SemiDisk will relieve your suffering.

Fast-acting.

The SemiDisk, a super-fast disk emulator, stores and retrieves data much faster than either a floppy or hard disk.

Easy to apply.

Installation is as easy as plugging the SemiDisk into an empty slot of your computer, and running the installation software provided.

Regular and extra-strength.

SemiDisk I is the standard model for S-100, SemiDisk II offers extra speed and flexibility for custom

S-100 applications.

Contains gentle buffers.

CP/M®80 installation software includes SemiSpool, which buffers print data in the SemiDisk. This allows the computer to be ready for other uses immediately after issuing a print command.

No emulator amnesia.

The optional Battery Backup Unit (BBU) plugs into the SemiDisk, and supplies power even when the computer is off. A battery keeps the data alive during power outages of four hours or more.

Stops head-aches.

Unlike a hard disk, which can 'crash' its head on the rotating disk

surface, and a floppy, which grinds the disk constantly, the SemiDisk gives you ultra-fast, silent data transfer.

And SemiDisk's price won't raise your blood pressure.

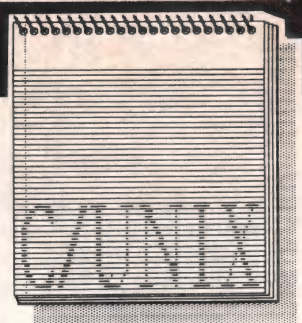
	512K	1Mbyte
SemiDisk I, S-100	\$995	\$1795
SemiDisk II, S-100	\$1245	\$2095
SemiDisk, TRS-80 II	\$995	\$1795
SemiDisk, IBM PC	\$945	\$1795
SemiDisk, Epson QX10	\$995	
SemiDisk, BBU	\$150	

SEMI DISK

SemiDisk Systems, Inc.
P.O. Box GG,
Beaverton, Oregon 97075
503-642-3100

Call 503-646-5510 for CBBS/NW and 503-775-4838 for CBBS/PCS, both SemiDisk-equipped computer bulletin boards (300/1200 baud). SemiDisk, SemiSpool trademarks of SemiDisk Systems. CP/M trademark of Digital Research

Circle no. 85 on reader service card.



by Anthony Skjellum

In this column, we'll consider some reader feedback on material presented in earlier columns. First, I will mention some new volumes available from the C User's Group. Following this, I'll present some corrections to errors in previous columns.

CUG Volumes

Machine-readable software can help you avoid a lot of frustration. To help readers of this column, I have created two C User's Group volumes: CUG *DDJ*, Volumes 1 and 2. They contain material up to and including the October 1984 column. The group's new address is:

C User's Group
415 East Euclid
McPhearson, KS 67460
(316) 241-5450

Volumes are available in several formats, including IBM PC and popular 5¼-inch CP/M-80 formats (e.g., Osborne DD). Contact Robert Ward at the above address for details.

Runge-Kutta Correction

Three errors were evident in the October column. The differential equation used in the example (from page 94) should have been

$$y'(t) = 1 + t - y$$

and not

$$y'(t) = 1 + y$$

as printed. Furthermore, the solution to this equation is

$$y(t) = t + 5.0 \cdot \exp(-t)$$

for the initial condition $y_0 = 5.0$.

Also, the `display()` function on page 98 of the October issue is corrected in Listing Three (page 102) of this issue.

X Grammar Examples

Some of the examples presented in the September column were set incorrectly. The errors principally involve the omission of semicolons at the ends of

certain lines. For example, in Figure 1, page 116, there should be a semicolon following the word `COMPLEX`. Furthermore, both assignment statements in the first `cadd()` function of Figure 2 are missing their semicolons. Figure 3 is missing its terminating brace. Finally, an errant semicolon appears on the first line of Figure 5.

Long Pointer Corrections

I presented a Long Pointer package in

the June column. Bruce Komusin wrote from Monaco to point out some errors in the assembly language routines. He writes:

"I just read your article in *DDJ* #92 about long pointers for C. I never [have] used C, but I know 8086 assembler. From your listing of `llsup.asm`, it is apparent that you overlooked the fact that the 8086 affects the flags when doing `INC` or `DEC` [instructions] for 16-bit [quantities]. This is a com-

```
ldec      proc near
           or          bx,bx
           jnz         ldec_1
           mov         ax,es
           sub         ax,1000h
           mov         es,ax
ldec_1:   dec         bx
           ret
ldec      endp
```

Figure 1

```
vec int sort_them(argcnt, argvec)
int argcnt;
int *argvec[]; /* Integer arguments */
{
    /*
     * exchange the values of the argvec array
     * here
     */
}
```

Figure 2

```
vec int sort_them(argcnt, argvec)
int argcnt;
int *(argvec[0])( );
int *argvec[]; /* The rest are integers */
{
    /* etc. */
}
```

Figure 3

mon error because it is different on the 8080. So, for example, you can save bytes and time in the routine line by removing the OR BX,BX."

While this concerns only inefficient coding, Mr. Komusin continues to point out a real bug:

"However, I really wrote this letter to warn you about ldec. Of course, it will not work as is because of the DEC BX changing the zero flag set up by the OR BX,BX. I suggest a change . . . that fixes everything."

The change is presented in Figure 1 (page 96). Beyond the basic fixes, Mr. Komusin suggests some increases in efficiency:

"However, here are some points about execution speed and byte efficiency. It is much faster to 'fall through' a conditional jump than to actually jump. So, if possible, it is always a good idea to arrange the code so that the normal case falls through and only the exceptional case jumps. As a side benefit, the exceptional case can then be shared."

A full set of improved routines are presented in Listing 1 (page 97). I want to thank Mr. Komusin for his letter and corrections.

Programming Philosophy

John A. Grosberg of Scottsdale, Arizona, wrote an interesting letter concerning programming style and philosophy. He wrote his letter after reading the August 1984 column, which included a short listing by Alex Cameron. Mr. Grosberg writes:

"Your column . . . caught my attention, particularly the short listing of Mr. Alex Cameron's routines to automatically allocate I/O buffers (page 119). I am writing to present a few ideas on program structure and will use his listing as an example. This is not an attack on his application or on the style he used in his listing—I assume that there were reasons for the form chosen. But my perfectionism was provoked by that listing, and the more I read it, the more I wanted to write.

"One important principle of program design is that the structure of the program (I will use *program*, *routine*, and *function* interchangeably for this discussion) should reflect the structure of the problem. This sounds nice, but what does it mean? Without guidelines

it is almost a theological principle, over which well-meaning people could argue loud and long and never come to agreement. The reason for this is that the 'structure of the problem' depends on one's viewpoint; i.e., it is relative to the observer . . . the program's structure reflects the way we are thinking about the problem."

Since the way we write programs is based on our viewpoint, Mr. Grosberg suggests a set of standard reference points:

"In mechanical drafting, there are three standard orthogonal viewpoints

that are used to describe most objects. They are called 'front,' 'side,' and 'top' views of the object. The structure of the physical object inheres in the spatial relationships of its elements, and these must be captured in the drawing.

"In software, an important aspect of structure is the temporal relationships among the elements. The two primary temporal relationships are sequence and frequency, and these relationships should be captured in the code. If one action occurs before another in time (sequence), then the first should precede the second in the code. If an action

```
vec char *dunno(argcnt, argvec)
int      argcnt;
char     *argvec[0];
long     *argvec[1];
double   *argvec[2];
COMPLEX  *argvec[3];
int      *argvec[];    /* rest are integers */

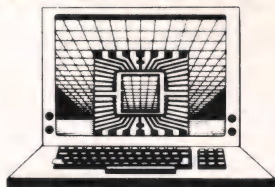
/*
 * Here we have a function whose first four arguments
 * are respectively: char, long, double, and COMPLEX
 * pointers. Anything after that is an integer
 */
}
```

Figure 4

```
;
; improvements to llsup routines by Bruce Komusin
; % Microworld
; L'Estoril
; 31 Ave. Princesse Grace
; Monte Carlo, Monaco
;
; these routines offer more temporal and byte-efficient
; code than those originally presented in llsup.asm
;
ldec      proc      near
or        bx,bx
jz        ldec_2
dec       bx
ret
ldec_2    dec       bx
ldec_3    mov       ax,es
sub       ax,1000h
mov       es,ax
ret
ldec      endp
lsub      proc      near
sub       bx,ax
jb        ldec_3
ret
lsub      endp
```

Listing One

"C" INTO THE FUTURE



WITH
db_VISTA

The first DBMS
designed exclusively
for the C language.

C is the applications development language chosen by many of the largest and most successful microcomputer software houses. Now with **db_VISTA**, C can be your development language choice, too.

db_VISTA is the database management system that helps you easily define and manage databases — no matter how complex your information structuring requirements. Features include:

- Written in C, under Unix.
- Minimal data redundancy using the network database model.
- Virtual memory disk accessing.
- Fast B*-tree indexing method for key files.
- Multiple key records—any or all data fields may be keys.
- Unlimited run-time distribution license available for \$795.
- Three month extended applications support included.
- PC-Write word processor/text editor included at no charge.

MONEY-BACK GUARANTEE

For Lattice C, DeSmet C, or Computer Innovations' C86 under MS-DOS, with thirty day money-back guarantee. Available soon for Unix/Fortune 32:16, Xenix/Altos 586, and CTOS/Convergent Technologies systems.

db_VISTA versions

Lattice	\$495
DeSmet	495
Computer Innovations	495
db_VISTA Manual	15

Development Packages:

Lattice C w/db_VISTA	\$795
Lattice C only	395
DeSmet C w/db_VISTA	595

RAINNA
CORPORATION
11717 Rainier Avenue South
Seattle, WA 98178
206/772-1515



occurs the same number of times (frequency) as another, then they should be in the same (logical) block of code."

I think that Mr. Grosberg's recommendations are practical. In my opinion, this type of coding technique could only improve maintainability of software. He continues:

"Expanding on the concept of temporal relationships as expressed in code, consider that on any single execution of a program, an element of that program may be executed once, more than once, or less than once [i.e., not executed]. If the element executes once and only once per program execu-

tion (sequence), it should simply be listed in sequence where it belongs. If the element executes more than once per execution (repetition), it should appear once in a loop. If the element executes less than once per program execution (alternation), it should appear once in a program branch statement. Finally, all elements that execute the same number of times should appear together in the listing."

While these points seem obvious to me (and also to Mr. Grosberg), it is clear that they are not often followed. I cannot claim to have adhered to these principles in the past, although I plan

```
#define NULL      0
#define BUFSIZ    256
#define TRUE      1
#define ERR       -1

slopen(filename, mode)
char *filename;
char *mode;
{
    int fd,
        err;
    if (fd = alloc(BUFSIZ))
    {
        switch(*mode)
        {
            case 'w': /* write mode */
                err = (fcreat(filename,fd) == ERR);
                break;
            case 'r': /* read mode */
                err = (fopen(filename,fd) == ERR);
                break;
            case 'a': /* append mode */
                err = (fappend(filename,fd) == ERR);
                break;
            default: /* invalid mode */
                err = TRUE;
                break;
        }
        if (err)
        {
            free(fd);
            fd = NULL;
        }
    }
    else /* alloc failure */
    {
        fd = NULL; /* redundant */
    }
    return(fd);
}
```

Listing Two

/* Revised <untested> version of "slopen." The original version was written by A. Cameron and published in DDJ, August 1984.

This revision is to illustrate the design principle that the structure of a program should reflect the behavior of the program (a corollary of the principle that the structure of a solution should reflect the structure of the problem).

The focus of this example is that the execution sequence and execution frequency are primary elements of problem structure and should be mirrored in the code.

by John A. Grosberg
[relevant for BDS C]

*/

Announcing a

TOTAL PARSER GENERATOR

<GOAL> :: = <RAPID> <COMPILER> <DESIGN>

SLICE YOUR COMPILER DEVELOPMENT TIME

An LR(1) parser generator and several sample compilers, all in Pascal for your microcomputer.

- Generates parser, lexical analyzer and skeleton semantics
- Universal, state-of-the-art error recovery system
- Adaptable to other languages
- Interactive debugging support
- Thorough documentation
- TURBO PASCAL™ INCLUDED FREE OF CHARGE
- Includes mini-Pascal compiler, assembler, simulator in SOURCE

SPECIAL INTRODUCTORY OFFER \$1995

OPARSER™ runs on IBM PC/DOS in Turbo Pascal. Parser generator in object form; all else in source. OPARSER takes a grammar and generates a correct, complete, high-performance compiler with skeleton semantics in Pascal source. Easy to add full semantics for YOUR application. Excellent for industrial and academic use. An accompanying textbook (SRA publishers) available in 1985. Training can be arranged. **Demo Disk available for \$50.00.**

Educational and quantity discounts available. Check, money order, Mastercard, Visa. California residents add 6.5% sales tax.

WRITE OR CALL FOR FREE BROCHURE.

Technical details: call 408/255-5574. Immediate delivery. CALL TODAY!

QCAD
SYSTEMS, INC.

1164 Hyde Ave., San Jose, CA 95129

TOLL FREE: 800-538-9787

(California residents call 408/255-5574)

™ Turbo Pascal is a registered trademark of Borland International.

Circle no. 76 on reader service card.

SMALL C FOR IBM-PC

Small-C Compiler Version 2.1 for PC-DOS/MS-DOS
Source Code included for Compiler & Library
New 8086 optimizations
Rich I/O & Standard Library

\$40

CBUG SOURCE LEVEL DEBUGGER FOR SMALL C

Break, Trace, and Change variables all on the source level
Source code included

\$40

Datalight

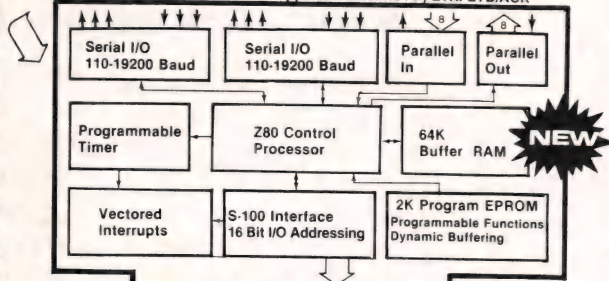
11557 8th Ave., N.E.
Seattle, Washington 98125
(206) 367-1803

ASM or MASM is required with compiler. Include disk size (160K/320K), and DOS version with order. VISA & MasterCard accepted. Include card no. & expiration date. Washington state residents include 9.9% sales tax. IBM-PC & PC-DOS are trademarks of International Business Machines. MS-DOS is a trademark of Microsoft Corporation.

Circle no. 31 on reader service card.

BUFFERED I/O BOARD

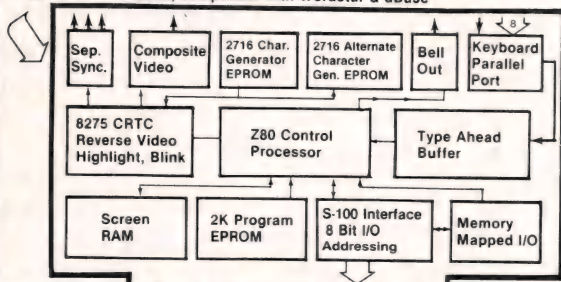
With despool functions, protocols supported: XON/XOFF, ETX: ETB/ACK



80 CHARACTER VIDEO BOARD

25 Lines with status, compatible with Wordstar & dBase

*\$49.50



Includes Bareboard, Heatsink & Documentation. Call or write for more information.



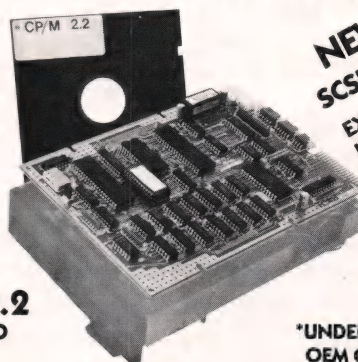
Simpliway Products Co.
P.O. Box 601
Hoffman Estates, IL 60195
(312) 359-7337



OEM dealer pricing available, \$3.00 S/H, IL. Res. add 7% tax.
dBase™ - of Ashton-Tate Corp. - Wordstar™ - of Micropro Int'l. Corp.

The Little Board™ ...\$349*

The world's simplest and least expensive CP/M computer



CP/M 2.2 INCLUDED

**NEW
SCSI/PLUS™
EXPANSION
I/O OPTION**

***UNDER \$200 IN
OEM QUANTITIES**

- 4 MHz Z80A CPU, 64K RAM, Z80A CTC, 2732 Boot ROM
- Mini/Micro Floppy controller (1-4 Drives, Single/Double Density, 1-2 sided, 40/80 track)
- Only 5.75 x 7.75 inches, mounts directly to a 5 1/4" floppy drive
- 2 RS232C Serial Ports (75-9600 baud & 75-38,400 baud), 1 Centronics Printer Port
- Power Requirement: +5VDC at .75A; +12VDC at .05A/On-board -12V converter
- CP/M 2.2 BDOS • ZCPR3 CCP • Enhanced AMPRO BIOS
- AMPRO Utilities included:
 - read/write to more than 2 dozen other formats (Kaypro, Televideo, IBM CP/M86....)
 - format disks for more than a dozen other computers
 - menu-based system customization
- BIOS and Utilities Source Code Available
- SCSI/PLUS Adapter:
 - Mounts directly to Little Board
 - Slave I/O board control
 - Full AHSC X3T9.2
 - 16 bidirectional I/O lines
 - \$99/Quantity 1

AMPRO
COMPUTERS, INCORPORATED

Distributor/Dealer/Reps
Inquiries Invited

Z80A is a registered trademark of Zilog, Inc.
CP/M is a registered trademark of Digital Research.

67 East Evelyn Ave. • Mountain View, CA 94041 • (415) 962-0230 • TELEX 4940302

Circle no. 87 on reader service card.

Circle no. 7 on reader service card.

Debugging Bugging You?

Torpedo program crashes and debugging delays with debugging dynamite for the IBM PC ...

UP PERISCOPE!

First, you install the hardware.

The hardware's a special memory board that fits in a PC expansion slot. Its 16K of write-protected memory contains Periscope's resident symbolic debugger. No runaway program, however berserk it may be, can touch this memory!

Then you UP PERISCOPE.

Use Periscope's push-button break-out switch to interrupt a running program ... even when the system's hung! Periscope supports Assembly, BASIC, C and Pascal. In addition to the usual debugging capabilities, some of Periscope's features are:

Stop your system in its tracks at any time.

Use symbol names instead of addresses.

Run a program on one monitor and debug on another.

Monitor your program's execution with Periscope's comprehensive breakpoints.

Debug memory-resident programs.

Put your time to better use.

The Periscope system is \$295. It carries a 30-day money-back guarantee and includes the memory board, remote break-out switch, debugger software, 100-page manual, and quick-reference card. The memory board is warranted for one year. A demonstration disk is \$5.00.

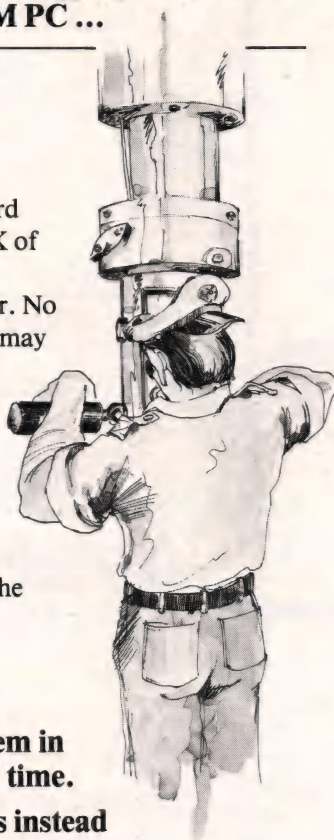
System requirements for Periscope are an IBM PC, XT or Compaq, PC-DOS, 64K RAM, 1 disk drive and an 80-column monitor. For MasterCard and Visa orders only, call 800/421-5300 (ext. R96) 24 hours a day. For additional information, call 404/256-3860 from 9 AM to 5 PM Eastern Time.

Get your programs up and running;

UP PERISCOPE!

Data Base Decisions / 14 Bonnie Lane / Atlanta, GA 30328

Circle no. 30 on reader service card.



to do so in the future. For those interested in pursuing the concepts further, he recommends *Practical LCP, a Direct Approach to Structured Programming*, by Albert C. Gardner (McGraw Hill, 1981). Mr. Grosberg has recoded Alex Cameron's listing to exemplify his comments; this code is presented in Listing Two (page 98). His comments concerning the code itself follow:

“... in Mr. Cameron's function 'sfopen,' the call to 'alloc' actually occurs only once per execution, but it is written three times in the code [sequence]. The 'return' occurs only once per execution, but is written 10 times [sequence]. The three main 'if' statements

```
if(*mode == 'x')
{
    ...
}
```

are written as if they occur sequentially, when in fact only one of them can occur per execution [alternation]. The structure of the code actually obscures the execution behavior of the function.”

Mr. Grosberg doesn't claim to have embodied his comments perfectly in Listing Two. He just created it (untested) to illustrate his remarks. I found the ideas worthwhile.

Another Response to the August Column

Mike Meyer writes the following concerning the August issue:

“I just read the August '84 column. Applause—you struck a solid, well-balanced position. If you can continue doing as well, you'll have a great column.

“Let me ask a favor: Please, PLEASE avoid the 'Unix/C is better/worse than <fill in blank> because <another blank>' type arguments. As you said, your column 'exists for discussing C and Unix as they are, with the problems they have.' Such discussions don't fit into that mold and tend to generate more heat than light.”

“Couple of nits: James Jones and Jeff Bowles are at uokvax, not ea. Their net address should be ucbvax!mtxinu!ea!uokvax!emjej, jab}. Finally, my last name is Meyer, not Meyers. If you could cease pluralizing me in the future, I'd appreciate it.”

I want to apologize for this error. Mr. Meyer has been a regular corre-

ASSEMBLE 3-6 TIMES FASTER ON THE IBM PC

Introducing **FAST ASSEM-86™**, the first Editor/Assembler for the IBM PC and PC compatibles. **FAST ASSEM-86™ (FASM)** is significantly faster and easier to use than the IBM Macro-Assembler (MASM). Whether you are new to assembly language and want to quickly write a small assembly language routine, or are an experienced MASM user tired of waiting months to assemble large files, **FAST ASSEM-86** will bring the excitement back to assembly language.

FAST ASSEM-86 IS MUCH FASTER:

- How fast is **FASM™**? The graph below shows relative assembly times for a 48K source file. For large files like this we blow MASM's doors off at 3 times their speed. For smaller 8K files we positively vaporize them at 6 times their speed.

FASM™ (110 sec.)

MASM v1.0 (340 sec.)

- FAST ASSEM-86** is faster for the following reasons: (1) Written entirely in assembly language (unlike MASM). (2) Editor, assembler and source file always in memory so you can go instantly from editing to assembling and back. (3) Eliminates the time needed to LINK programs. Executable .COM files can be created directly. (Also creates .OBJ files completely compatible with the IBM linker).

FAST ASSEM-86 IS EASIER TO USE:

FASM includes many other features to make your programming simpler.

- Listings are sent directly to screen or printer. Assemblies can be single stepped and examined without having to leave the editor.
- Access the built in cross reference utility from the editor.
- Full support of 186 and 286 (real mode) instructions.
- Both Microsoft and 8087 floating point formats are supported. 8087 and 287 instructions supported directly without macros for faster assembly.
- Calculator mode: Do math in any radix even using symbols from the symbol table.
- Direct to memory assembly feature lets you test execute your code from editor.
- Coming soon: A coordinated symbolic debugger.

COMPATIBILITY: **FASM** is designed for source code compatibility with MASM and supports most of its important features.

Introductory Price \$199

Speedware™

IBM, Microsoft trademarks of IBM Corp., Microsoft Corp. respectively.

Dealer inquiries welcome

916-966-6247

Box D1, 2931 Northrop Avenue
Sacramento, CA 95825

Circle no. 97 on reader service card.

GREAT PROGRAM . . . TERRIBLE MANUAL . . .

How often have you seen or heard that said about software?

If you have a first-class program you deserve the **BEST** documentation. At A/N SOFTWARE we can provide you with a top-quality manual in the shortest possible time.

Whether it's a complete service from writing, to design, artwork, typesetting, mechanicals, and printing, or any step along the way, your manual will be in a class with the best documentation available today.

Our staff consists of writers, artists, testers, and even includes a CPA. Business and technical programs are a specialty. We pride ourselves on designing manuals that look expensive, but aren't.

We understand the time constraints of the software business; we'll meet your deadlines.

We're proud of our work and would like to send you samples. For additional information, samples, or just some advice from the experts call:

. . . 516-549-4090 . . . or write



SOFTWARE Inc.

P.O. Box 895

Melville, NY 11747

Circle no. 2 on reader service card.

GTEK INC.

EPROM PROGRAMMER

DEVELOPMENT HARDWARE/SOFTWARE

HIGH PERFORMANCE/ COST RATIO

(601) 467-8048

Compatible w/all Rs 232 serial interface port • Auto select baud rate • With or without handshaking • Bidirectional Xon/Xoff and CTS/DTR supported • Read pin compatible ROMS • No personality modules • Intel, Motorola, MCS86, Hex formats • Split facility for 16 bit data paths • Read, program, formatted list commands • Interrupt driven, program and verify real time while sending data • Program single byte, block, or whole EPROM • Intelligent diagnostics discern bad and erasable EPROM • Verify erasure and compare commands • Busy light • Complete w/Textool zero insertion force socket and integral 120 VAC power (240 VAC/50Hz available)

DR Utility Package allows communication with 7128, 7228, and 7956 programmers from the CP/M command line. Source Code is provided. PGX utility package allows the same thing, but will also allow you to specify a range of addresses to send to the programmer. Verify, set the Eprom type.

MODEL 7316 PAL PROGRAMMER

Programs all series 20 PALS. Software included for compiling PAL source codes.

Software Available for CPM,¹ ISIS,² TRSDOS,³ MSDOS.⁴

1. TM of Digital Research Corp.
2. TM of Intel Corp.
3. TM of Tandy Corp.
4. TM of Microsoft.

Post Office Box 289
Waveland, Mississippi 39576
(601)-467-8048

Avocet Cross Assemblers are available to handle 8748, 8751, Z8, 6502, 680X, etc. Available for CP/M and MSDOS computers. Order by processor type and specify kind of computer.

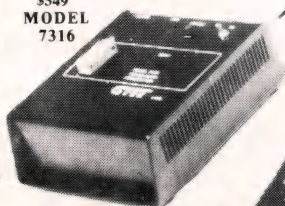
Model DE-4 U/V Products hold 8, 28 pin parts. High quality professional construction.

\$879 stand alone
MODEL 7956

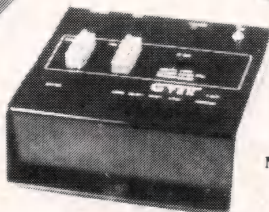


MODEL 7956
GANG PROGRAMMER
Intelligent algorithm. Stand alone, copies eight EPROMS at a time. With RS-232 option \$1099.

\$549
MODEL
7316



\$1195
MODEL
7324



MODEL 7324 PAL PROGRAMMER
Programs all series 20 & 24 PALS. Operates stand alone or via RS232.

\$549
MODEL
7228



MODEL 7228
EPROM PROGRAMMER
All features of Model 7128 plus Auto Select Baud . . . super fast adaptive programming algorithms. low profile aluminum enclosure. Programs 2764 in one minute!

\$429
MODEL
7128



MODEL 7128 EPROM PROGRAMMER
Programs and Read:

NMOS	NMOS	CMOS	EEPROM	MPU'S
2508	2758	27C16	5213	8748
2516	2716	27C32	5213H	8748H
2532	2732	C6716	X2816	8749H
2564	2732A	27C54	48016	8741
68766	2764		12816A	8742H
68764	27128			8741H
8755	27256			8751
5133				

Model 7128-L1,L2,L2A	\$239.00
Model 7128-24	\$329.00
DR8 or DR5	\$ 30.00
DR8PGX or DR5PGX	\$ 75.00
Cross Assemblers	\$200.00
XASM (for MSDOS)	\$250.00
U/V Eraser DE-4	\$ 78.00
RS232 Cables	\$ 30.00
8751 adapter	\$174.00
8755 adapter	\$135.00
48 Family adapter	\$ 98.00

Circle no. 42 on reader service card.

spondent, which means I had ample opportunity to see his name and reproduce it properly. (I'll get his name straight from now on.) He continues:

"To add some constructive comment, I'd like to point out that relying on library utilities for things does not guarantee portability. For instance, many C implementations won't have the Unix math(3) library or the qsort(3) routines. Of note is that the current AT&T Unix distribution doesn't include the dbm(3) routines from Unix version 7. I use those routines to fix the 'everything is line-oriented ASCII' problem with Unix, and some of the AT&T sites that don't have that library complained when they got copies of my software."

This is an interesting point that I had not considered. It adds more complexity to the idea of C/Unix software portability. What libraries can and cannot be assumed when writing a program? Is it OK to think of libraries such as CURSES as standard?

Comments on the X Grammar

I received several comments about the X grammar. In this column, I present one letter; the rest are reserved for the February 1985 column [DDJ No. 100]. John M. Gamble of Batavia, Ohio, writes:

"Your column on extensions to the C language was very interesting. I have a few comments.

"(1) To keep analogy between functions and ops, I think that it should be legal to declare static ops."

This sounds fine, but what is a static

oper? Since I'm not sure what Mr. Gamble means, I can't really comment. He continues:

"(2) I have trouble thinking of any justification for adding one more reserved word (loop) just to do what 'for(;;)' does just as well. If it really offends your eye, couldn't you just use #define to substitute for it?"

I agree. I only mentioned "loop" because I wanted an efficient way to specify an unconditional loop. This is fine since "for(;;)" shouldn't produce unnecessary instructions in object code. Mr. Gamble continues his list of comments as follows:

"(3) I think that your method of declaring argument lists in vec functions is too limited to be practical. A function list is not analogous to argv, which deals only with character strings. A function, after all, deals with all sorts of variables. To get around this problem, I have thought of two possible solutions:

"(a) Require that the first argument be a string equivalent to printf's control string. Quite frankly, I dislike this solution. Deciphering the control string would be a pain, and the code needed to deal with this pain would probably ruin C's reputation for compact code.

"(b) Declare the types of the argument list members in the function itself. This would be efficient and easy to modify later on. For example, say that you wish to have some integer variables, and you wish to exchange their values so that they are in [numerical] order. Rather than going through the trouble of inserting the values in an

array, calling a sorting routine, and recovering the values from the array, a vec function called sort_them() might be easier to use. The declaration might be as depicted in Figure 2 (page 96).

"If you wanted to make the function more flexible by allowing the ordering to be user-specified, we could have the first argument be a function. Then the declaration would resemble Figure 3 (page 96).

"Of course, we are not limited to integer pointers. A vec function could just as easily have arguments of all sorts. Such an example is presented in Figure 4 (page 97).

"Since the argvec array consists of pointers only, the addresses of the argument list are passed in, not the values. Therefore, register variables may not be used in the list. Also, since passing addresses is the default, we can drop the '&' before each variable [in the calling sequence]."

I think Mr. Gamble's ideas are valid and are consistent with my original intentions for an extended grammar. Does anyone else have further comments about vec functions?

Conclusions

I have wound up the year by including some corrections and comments about previous columns. The programming structure comments offered by Mr. Grosberg were particularly interesting to me, and I hope that others find them useful as well. I was pleased with Mr. Gamble's suggestions concerning C extensions, and I look forward to additional remarks in this area.

My next column will appear in the February 1985 DDJ. This is issue No. 100, a landmark for the journal. In this column, I'll include additional reader feedback and suggestions as well as possible topics for future columns. Have a happy holiday season.

DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 198.

```
/* display( ): subroutine to print an ascii file on the console */
display(fname)
char *fname;
{
    char c; /* character to output */
    FILE *disp;
    if((disp = fopen(fname, "r")) == NULL)
        return(-1); /* can't open file */
    while((c = getc(disp)) != EOF) /* print the file */
    {
#ifdef UNIX
        if(c == TEOF) /* text end of file */
            break;
#endif
        putchar(c & 127); /* output each character less parity */
    }
    fclose(disp); /* close the file */
    return(0); /* successful completion */
}
```

Listing Three

Thanks to YOU We're Growing
with YOU and your Computer . . .



LEO ELECTRONICS, INC.
P.O. Box 11307
Torrance, CA. 90510-1307
Tel: 213/212-6133 800/421-9565
TLX: 291 985 LEO UR

**We Offer . . . PRICE . . . QUALITY . . .
PERSONAL SERVICE**

64K UPGRADE

9 Bank (IBM PC)	\$43.65	(150ns)		
	\$41.85	(200ns)		
4164 (150ns)	\$4.85 ea.			
(200ns)	\$4.65 ea.			
8 Bank (other PC)	\$38.80	(150ns)		
	\$37.20	(200ns)		
4164 (150ns)	\$4.85 ea.			
(200ns)	\$4.65 ea.			
256K "Mother-Saver" Upgrade				
256K - (150ns)	\$36.00 ea.			
6116P-3 -	\$4.40	2732 -	\$3.95	
2716 -	\$3.20	2764 -	\$7.00	
TMS-2716 -	\$4.95	27128 -	\$24.00	

We accept checks, Visa, Mastercard or Purchase Orders from qualified firms and institutions. U.S. Funds only. Call for C.O.D. California residents add 6 1/2% tax. Shipping is UPS. Add \$2.00 for ground and \$5.00 for air. All major manufacturers. All parts 100% guaranteed. Pricing subject to change without notice.

Circle no. 59 on reader service card.

Now Your Computer Can See! \$295.00*

A total imaging system complete and ready for plug-and-go operation with your personal computer.

The MicronEye™ offers selectable resolution modes of 256 x 128 and 128 x 64 with operating speeds up to 15 FPS. An electronic shutter is easily controlled by software or manual functions, and the included sample programs allow you to continuously scan, freeze frame, frame store, frame compare, print and produce pictures in shades of grey from the moment you begin operation.

Only the MicronEye™ uses the revolutionary IS32 OpticRAM™ image sensor for automatic solid state image digitizing, with capability for grey-tone imaging through multiple scans. And with these features, the MicronEye™ is perfectly suited for graphics input, robotics, text and pattern recognition, security, digitizing, automated process control and many other applications.

The MicronEye™ is available with immediate delivery for these computers: Apple II, IBM PC, Commodore 64 and the TRS-80CC (trademarks of Apple Computer Inc., International Business Machines, Commodore Corp., and Tandy Corp. respectively).

Phone for MicronEye™ information on the Macintosh, TI PC and RS232 (trademarks of Apple Computer Inc. and Texas Instruments respectively).

* (Add \$10.00 for shipping and handling [Federal Express Standard Air]; residents of the following states must add sales tax: AK, AZ, CA, CO, CT, FL, GA, IA, ID, IL, IN, LA, MA, MD, ME, MI, MN, NC, NE, NJ, NY, OH, PA, SC, TN, TX, UT, VA, VT, WA, WI.)



MicronEye™
"Bullet"

**MICRON
TECHNOLOGY, INC.**

VISION SYSTEMS
2805 East Columbia Road
Boise, Idaho 83706
(208) 383-4106
TWX 910-970-5973

Circle no. 63 on reader service card.

ICs PROMPT DELIVERY!!! SAME DAY SHIPPING (USUALLY)

DYNAMIC RAM

256K	256Kx1	120 ns	\$24.47
256K	256Kx1	150 ns	\$22.47
64K	64Kx1	120 ns	3.87
64K	64Kx1	150 ns	3.67
64K	64Kx1	200 ns	3.56

EPROM

27256	32Kx8	300 ns	\$45.97
27128	16Kx8	300 ns	13.67
27C64	8Kx8	200 ns	22.50
2764	8Kx8	250 ns	6.50
2732	4Kx8	250 ns	6.37
2716	2Kx8	450 ns	3.50

STATIC RAM

6264P	8Kx8	150 ns	\$23.67
6116P	2Kx8	150 ns	4.37

QUANTITY ONE PRICES SHOWN

Open 6 1/2 days: We can ship via Fed-Ex on Sat.

MasterCard/VISA or UPS CASH COD

Factory New, Prime Parts

µP∞

MICROPROCESSORS UNLIMITED

24,000 South Peoria Ave. (918) 267-4961
BEGGS, OK. 74421

Prices shown above are for November 3, 1984

Please call for current prices & volume discount. Prices subject to change. Please expect higher prices on some parts due to world wide shortages. Shipping and insurance extra. Cash discount prices shown. Small orders received by 6 PM CST can usually be delivered to you by the next morning, via Federal Express Standard Air @ \$6.75!

Circle no. 64 on reader service card.

OPT-TECH SORT™

SORT/MERGE program for IBM-PC & XT

Now also sorts dBASE II files!

- Written in assembly language for **high performance**
Example: 4,000 records of 128 bytes sorted to give key & pointer file in 30 seconds. **COMPARE!**
- Sort ascending or descending on up to nine fields
- Ten input files may be sorted or merged at one time
- Handles variable and fixed length records
- Supports all common data types
- Filesize limited only by your disk space
- Dynamically allocates memory and work files
- Output file can be full records, keys or pointers
- Can be run from keyboard or as a batch command
- Can be called as a subroutine to many languages
- Easy to use, includes on-line help feature
- Full documentation — sized like your PC manuals
- **\$99** — VISA, M/C, Check, Money Order, COD, or PO
Quantity discounts and OEM licensing available

To order or to receive additional information write or call:

OPT-TECH DATA PROCESSING

P.O. Box 2167 Humble, Texas 77347
(713) 454-7428

Requires DOS, 64K and One Disk Drive

Circle no. 68 on reader service card.

Put Dr. Dobb's Under the Christmas Tree!
Treat your friend (or yourself!) to Dr. Dobb's Journal

BOUND VOLUME 7

Every 1982 Issue Available For Your Personal Reference.

Vol. 7 1982

In 1982 we introduced several significant pieces of software, including the RED text editor and the Runic extensible compiler, and we continued to publish utility programs and useful algorithms. Two new columns, The CP/M Exchange and The 16-Bit Software Toolbox, were launched, and we devoted special issues to FORTH and telecommunications. Resident Intern Dave Cortesi

supplied a year of "Clinic" columns while delivering his famous review of JRT Pascal and writing the first serious technical comparison of CP/M-86 and MSDOS. This was also the year we began looking forward to today's generation of microprocessors and operating systems, publishing software for the Motorola 68000 and the Zilog Z8000 as well as Unix code. And in December, we looked beyond, in the provocative essay, "Fifth-generation Computers."

Vol. 1 1976

The material brought together in this volume chronicles the development in 1976 of Tiny BASIC as an alternative to the "finger blistering" front-panel, machine-language programming which was then the only way to do things. This is always pertinent for bit crunching and byte saving, language design theory, home-brew computer construction and the technical history of personal computing.

Topics include: Tiny BASIC, the (very) first word on CP/M, Speech Synthesis, Floating Point Routines, Timer Routines, Building an IMSAI, and more.

Vol. 2 1977

1977 found DDJ still on the forefront. These issues offer refinements of Tiny BASIC, plus then state-of-the-art utilities, the advent of PILOT for microcomputers and a great deal of material centering around the Intel 8080, including a complete operating system. Products just becoming available for reviews were the H-8, KIM-1, MITS BASIC, Poly Basic, and NIBL.

Articles are about Lawrence Livermore Lab's BASIC, Alpha-Micro, String Handling, Cyphers, High Speed Interaction, I/O, Tiny Pilot & Turtle Graphics, many utilities, and even more.

Vol. 3 1978

The microcomputer industry entered its adolescence in 1978. This volume brings together the issues which began dealing with the 6502, with mass-market machines and languages to match. The authors began speaking more in terms of technique, rather than of specific implementations; because of this, they were able to continue laying the groundwork industry would follow. These articles relate very closely to what is generally available today.

Languages covered in depth were SAM76, Pilot, Pascal, and Lisp, in addition to RAM Testers, S-100 Bus Standard Proposal, Disassemblers, Editors, and much, much more.

Vol. 4 1979

This volume heralds a wider interest in telecommunications, in algorithms, and in faster, more powerful utilities and languages. Innovation is still present in every page, and more attention is paid to the best ways to use the processors which have proven longevity—primarily the 8080/Z80, 6502, and 6800. The subject matter is invaluable both as a learning tool and as a frequent source of reference.

Main subjects include: Programming Problems/Solutions, Pascal, Information Network Proposal, Floating Point Arithmetic, 8-bit to 16-bit Conversion, Pseudo-random Sequences, and Interfacing a Micro to a Mainframe—more than ever!

Vol. 5 1980

All the ground-breaking issues from 1980 in one volume! Systems software reached a new level with the advent of CP/M, chronicled herein by Gary Kildall and others (DDJ's all-CP/M issue sold out within weeks of publication). Software portability became a topic of greater import, and DDJ published Ron Cain's immediately famous Small-C compiler—reprinted here in full!

Contents include: The Evolution of CP/M, a CP/M-Flavored C Interpreter, Ron Cain's C Compiler for the 8080, Further with Tiny BASIC, a Syntax-Oriented Compiler Writing Language, CP/M to UCSD Pascal File Conversion, Run-time Library for the Small-C Compiler and, as always, even more!

Vol. 6 1981

1981 saw our first all-FORTH issue (now sold out), along with continuing coverage of CP/M, small-C, telecommunications, and new languages. Dave Cortesi opened "Dr. Dobb's Clinic" in 1981, beginning one of the magazine's most popular features.

Highlights: information on PCNET, the Conference Tree, and The Electric Phone Book, writing your own compiler, a systems programming language, and Tiny BASIC for the 6809.

YES!

- ☐ Please send me the following Volumes of **Dr. Dobb's Journal**.
☐ ALL 7 for ONLY \$165, a savings of over 15%!

Payment must accompany your order.

Please charge my: ☐ Visa ☐ MasterCard ☐ American Express
I enclose ☐ Check/money order

Card # _____ Expiration Date _____

Signature _____

Name _____ Address _____

City _____ State _____ Zip _____

Mail to: Dr. Dobb's Journal, 2464 Embarcadero Way, Palo Alto, CA 94303

Allow 6-9 weeks for delivery.

Circle no. 124 on reader service card.

This offer expires February 28, 1985

Vol. 1	_____ x	\$26.75 =	_____
Vol. 2	_____ x	\$27.75 =	_____
Vol. 3	_____ x	\$27.75 =	_____
Vol. 4	_____ x	\$27.75 =	_____
Vol. 5	_____ x	\$27.75 =	_____
Vol. 6	_____ x	\$27.75 =	_____
Vol. 7	_____ x	\$30.75 =	_____
All 7	_____ x	\$165.00 =	_____

Sub-total \$ _____

Postage & Handling _____
Must be included with order.

Please add \$1.25 per book in U.S.
(\$3.25 each surface mail
outside U.S. Foreign Airmail
rates available on request.)

TOTAL \$ _____

B. G. MICRO

P. O. Box 280298 Dallas, Texas 75228
(214) 271-5546



Big Computer Mfg. Makes \$900,000 Goof!!

COMPUTER/DISK DRIVE SWITCHING POWER SUPPLY

ORIGINAL
OEM COST
\$72 EACH!

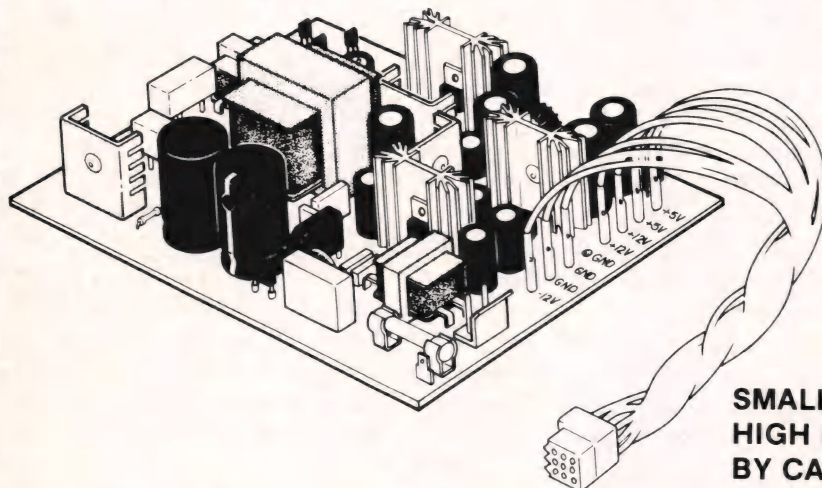
ORIGINALLY DESIGNED TO RUN A Z-80
BASED SINGLE BOARD COMPUTER
WITH TWO 5-1/4 IN. DISK DRIVES AND
CRT MONITOR.

BRAND NEW: UNUSED!

\$37⁵⁰ EA

3 FOR \$95⁰⁰

ADD \$1.50 PER UNIT FOR UPS



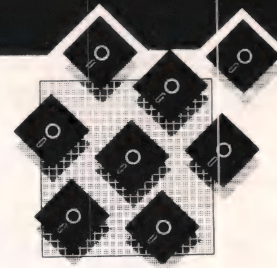
SPECS: + 5VDC 5 AMPS MAX
#1 + 12 VDC 2.8 AMPS MAX
#2 + 12 VDC 2.0 AMPS MAX
- 12 VDC .5 AMPS MAX

INPUT: 115 or 230 VAC 60Hz

SMALL SIZE: 6-1/8 x 7-3/8 In.
HIGH EFFICIENCY SWITCHER MFG.
BY CAL. DC IN USA!

The poor Purchasing Agent bought about 10 times as many of these DC switchers as his company would ever use! We were told that even in 10,000 piece lots they paid over \$72 each for these multi-output switchers. When this large computer manufacturer discontinued their Z-80 Computer, guess what the Big Boss found in the back warehouse; several truckloads of unused \$72.00 power supplies. Fortunately we heard about the deal and made the surplus buy of the decade. Even though we bought a huge quantity, please order early to avoid disappointment. Please do not confuse these high quality American made power supplies with the cheap import units sold by others.

TERMS: Orders over \$50 add 85¢ insurance. No COD. Tex. Res. Add 6% Sales Tax. Subject to prior sale. Foreign orders: US funds only. We cannot ship to Mexico. Foreign countries other than Canada add \$6 per board shipping.



NCI COHERENT

Company: Network Consulting Inc., Suite 110, 3700 Gilmore Way, Burnaby, B.C. Canada V5G4M1

Computer: IBM PC and IBM XT

Price: \$695

Circle Reader Service No. 127

Reviewed by A. Gomez

COHERENT was developed by the Mark Williams Co. as a Unix look-alike that did not have to pay royalty fees to AT&T. NCI has modified COHERENT with a number of real-time enhancements to enable efficient multiuser operation with small machines. The kernel is ROMable and may be embedded in peripheral modules and software for communications packages. The utilities for the most part were rewritten in assembly language for speed of execution. Finally, specific hardware is supported that allows expedient processing in a Unix style environment.

Features

The features of NCI COHERENT fall into two parts: hardware and software support. Hardware support is the support of a specific item of hardware that allows flexibility of configuration or enhances the performance of the system. Software features are those that provide ease of programming or system use. By this definition, a hardware feature may be a piece of software (e.g., a device driver).

The hardware features of NCI COHERENT are:

- Choice of hard disk support: NCI COHERENT provides device drivers for the XT, CORVUS, CORONA, DAVONG, TECMAR, GENIE, EAGLE, INTERPHASE SMD, MAYNARD SASI interface, and COLUMBIA disk controllers and drives. Note that this feature allows NCI COHERENT to operate on compatibles as

well as the PC itself.

- Ziatech IEEE-488 interface: The 488 interface is important in control applications.
- Interactive Data Systems 1600 BPI tape drive interface: This device driver allows the use of 9-track tape with NCI COHERENT.
- Tall Tree Systems JRAM card: This card is frequently used as a RAM disk because of its ability to window itself in an already RAM-filled PC. It has 512K of RAM on each card and can increase performance of swapping systems by being part of the file system.
- Persyst 4-line serial port card with onboard 8088: This allows the main processor to be off-loaded and permits the suitable execution of real-time applications in a multiuser environment.
- Control Systems Hostess 8-line serial port card: This card is necessary to provide the 11-user capacity that is claimed by NCI. The card supports XON/XOFF and variable size words (5 to 8 bits) in either interrupt or polled mode.
- Hayes autodialer (or any other auto-

dialer with programming effort): This provides support for the "cu" program.

- Support of the AST real-time clock via the "clock" command.
- Support of three parallel line printers.

The software features of NCI COHERENT are:

- Kernel parameters: NCI COHERENT provides the ability to change the kernel parameters via commands (e.g., partitioning of RAM disk).
- TERMCAP: This package will allow programs that are screen based to be terminal independent. It is widely used in the Unix community.
- 165 Unix V7 commands: These commands are rewritten in assembly language for speed.
- Lex and Yacc: A lexical analyzer and compiler compiler, commonly found and used on Unix systems, are of great value to developers who need to create parsers and lexical analyzers.
- RCS: This is a source code control system.
- Unix SYSTEM V memory routines: These routines are supported in the kernel and include memccpy,

Benchmark	Compile Time (sec)	Execution Time (sec)		
		Real	User	System
Pipes	30	27.4	0.1	22.0
System call	19	29.5	2.8	26.4
Function call	18	0.5	0.3	0.1
Sieve	19.6	8.5	8.1	0.2
Disk write	32.1	7.8	0.1	6.3
Disk read	33.0	16.0	0.1	7.0
Shell	-	13.0	1.4	5.3
Loop	18.2	26.6	26.4	0.1

Multiprocess benchmark	# Processes					
	(1)	(2)	(3)	(4)	(5)	(6)
Multi.sh	15.1	21.3	29.1	36.6	45.8	52.1

Table 1

memchr, memcmp, memcpy, and memset.

- Prolog interpreter: The prolog interpreter is frequently used in artificial intelligence work and control applications.
- An IEEE floating point math package that is accessible by the Unix math library routines.
- A screen editor ("see"): This is not the VI or EMACS, with which all of us are familiar, but a screen editor.
- License fees: The single-user price is the multiuser price; no additional fees are necessary. Furthermore, runtime only versions are available for applications with embedded kernels and offboard processors. This allows software developers to use the facilities of the NCI COHERENT kernel in their product without having the customer pay the full price of NCI COHERENT.
- Coming soon is a MSDOS bridge that allows a MSDOS program to run under NCI COHERENT.

Documentation consists of a reference manual (sections 1–8 of the Unix manual), a tutorial manual (ed, m4, lex, rcs, etc.), and the book *Using the UNIX System* by Richard Gauthier. The documentation was complete and organized in a manner consistent with other Unix systems.

Unix Compatibility

NCI COHERENT was designed to be compatible with Unix V7, and it looks like NCI has achieved its goals. All the system calls are identical, the libraries are identical, and only minor functions or user programs are different.

To check its compatibility, I moved various applications between a V7 Unix, the NCI COHERENT, and a SYSTEM V Unix. Without exception, all worked without error. Of course, these programs did not use the additional features of SYSTEM V, but then I was checking for V7 compatibility.

Benchmarks

Nothing is more controversial than benchmarks in product marketing. It is really hard to choose a set of programs that presents a fair comparison of different products. The benchmarks that I used in measuring NCI COHERENT were those found in the July 1984 issue of *Byte* magazine in an article named

"Benchmarking UNIX Systems" by David F. Hinnant. These benchmarks appear to be fair and, more importantly, to provide a set of measurements upon which to grade the performance of NCI COHERENT. In that article, Mr. Hinnant provides timings for nine benchmarks in 15 different systems. Table 1 (on page 106) shows the measurements of NCI COHERENT for the same benchmarks.

By comparing NCI COHERENT to other Unix systems on the PC, we see no improvement in the single-user environment. However, as the number of active processes increases, the degradation of NCI COHERENT is less than other Unix systems.

Closing Notes

In general, I was impressed by NCI COHERENT. NCI has placed its emphasis on the performance of a system with more processes than that found in a single-user environment. Although I did not test the multiuser capability of the product, I do not doubt that it exists because of the orientation of the kernel. The price is reasonable, considering other offerings for the IBM PC, and the company looks like it's heading toward full compatibility with Unix SYSTEM V, which allows a future for its products and support.

Turbo Pascal Version 2.0

Company: Borland International,
4113 Scotts Valley Drive,
Scotts Valley, CA 95066, 1-
800-227-2400 ext.968 out-
side California, 1-800-772-
2666 ext.968 in California

Computer: MSDOS, PCDOS

Price: \$49.95

\$89.95 with 8087 support

Circle Reader Service No. 129

Reviewed by Karl R. Kachigan

Well, it just had to happen. Friends were talking about it, the computer shows featured it, and just about every magazine beamed about its significant performance for the price. Just what kind of decent Pascal could this be for \$49.95? Originally, because it didn't support graphics on my IBM PC, I ignored it, continuing to use my UCSD p-System Pascal with its Turtlegraphics and other goodies. Then version 2.0 was announced—it supported graph-

ics, color, sound, windows, overlays, and was still only \$49.95. I nibbled. I telephoned in my order, and about a week later I became a believer. For the price and its features, Turbo Pascal compares quite well with my UCSD p-System Pascal.

Before you wonder how I was converted, let me say that Turbo Pascal is truly a good solution for both the novice and the experienced programmer. When Turbo Pascal is properly installed, and that isn't a difficult task, you have a nicely integrated programming system. The editor/compiler lets you compose a program, compile it, and either run it or return to the editor to correct errors flagged by the system, all without using the disk.

To those unfamiliar with the UCSD p-System, its editor, filer, and compiler are integrated but usually not simultaneously resident in memory (unless you configure a RAM disk that can simulate this). The UCSD Pascal compiler/editor obviously was the model for Turbo Pascal. Both indicate syntax errors with descriptive messages and will point to them if you return to the editor. Turbo Pascal just seems faster and slicker at it; plus, for those of us using the IBM PC, it runs under PCDOS. Only recently has the UCSD p-System been capable of doing this. However, the UCSD p-System does provide more capability, machine control, and transportability—but at a much higher cost.

This review should give you a better feel for the version 2.0 enhancements, especially on the IBM PC. I will build upon David Clark's review of Version 1.0 in the June 1984 issue of *DDJ*. Please reread this article if some of my points seem unclear. Because many of the fancy additions (the manual calls them "IBM PC goodies") are specific to the IBM PC and its compatibles, I will try to point out what features are applicable to all machines and which are IBM specific. Table 2 (page 110) lists all of the enhancements in version 2.0.

What Is Turbo Pascal?

Turbo Pascal is distributed as either one or two disks (the second is for MSDOS/PCDOS 8087 support), two manuals, several programs including sample programs, and an update file giving last minute information. Turbo

Pascal itself is composed of either two or three files: compiler/editor (TURBO.COM, TURBO.CMD, TURBO-87.COM, or TURBO-87.CMD), error message data file (TURBO.MSG), and for CP/M-80 only an overlay file (TURBO.OVR). A terminal installation program, provided to customize the Turbo Pascal screen control and editor features, includes the program itself (TINST.COM or TINST.CMD), a message data file (TINST.MSG), and a terminal data file (TINST.DTA, not on the IBM PC version). Provided on all versions is a program lister (TLIST.COM or TLIST.CMD) and a sample spreadsheet program called MicroCalc (supplied in source code form). Included with the MSDOS/PCDOS versions are information files on using MSDOS function calls, external assembly language routines, and interrupts, each with comments and a sample program listing. The IBM PC version adds sam-

ple programs on color, graphics, sound, and windows—all in source code format.

The manuals, an updated version 1.0 manual and a short version 2.0 supplement, attempt to cover all versions of Turbo Pascal (CP/M-80, MSDOS, PCDOS, and CP/M-86). They are typeset, seemingly well structured, and indexed.

Standard Turbo Pascal uses real numbers with a range of $1.0E-38$ to $1.0E+38$ and 11 significant digits. With the 8087 support, these expand to $4.19E-307$ to $1.67E+308$ and 16 significant digits. As for RAM requirements, version 2.0 for MSDOS takes 35K and for PCDOS 36K, compared with 33K for the old version 1.0 16-bit implementations.

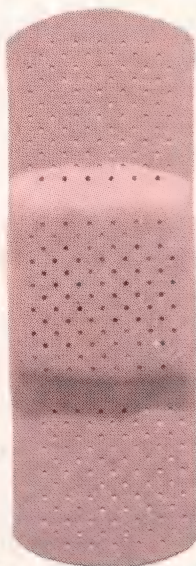
Editor Enhancements

I've mentioned that the built-in editor is a nice feature. A full screen editor, it

requires specific information on your terminal's control sequences and editing commands/keystrokes. Here is where the terminal installation program is helpful. It has two modes: select (or define) the terminal used and define the edit command keystrokes. For the non-IBM versions, a large selection of the most popular terminals are listed. If you are using one on the list, just select it. If yours is not listed, you can add your terminal to the list by entering the appropriate information, such as the cursor addressing, screen clear, insert/delete line, and enhance on/off control sequences. In either case, Turbo Pascal is now installed for the selected terminal.

I installed the MSDOS version on my HP-150 by adding my terminal to the list and found the installation program quite accommodating. It was gratifying to see a program that could adapt to my HP-150's lengthy cursor ad-

dBASE III



VS.

You waited years
for an advanced version of dBASE II.
Without the bugs.
Without the limitations.
It never came.

Instead, you got dBASE: III.
A half solution.
A bandage instead of a cure,
so to speak.

Here's what we mean.

As an applications programmer,
you're now supposed to use
dBASE III to write a program on
single-user 16-bit PCs...use dBASE II
to write the same application for
8-bit machines and use heaven
knows what to handle the multi-user
or networked situations.

Contrast that with Q-PRO 4...the
true 4th generation applications
development language for micro-
computers.

*Don't let anybody
do a number on you.*

CCP/M, CP/M, and MP/M are trademarks of Digital Research. TurboDOS, MmmOST, MUSE, NSTAR, MS-DOS, PC-DOS, PC Net, DNA, EtherShare, and NetWare are trademarks of System 2000.

addressing sequence (typically nine characters with ASCII row and column addressing). The IBM PC version lets you select from six display modes: default monitor, monochrome monitor, mono or color 40 columns, and mono or color 80 columns.

As for the editor, 45 commands are listed, almost all of them standard WordStar features. The version 2.0 manual claims to have added seven new commands, but they were also listed in the version 1.0 manual. (Apparently these were documented before they were actually implemented.) I felt right at home using the familiar WordStar control sequences again. This is truly a nice touch, especially having the same block move/copy/delete and find/search/replace capabilities. Not only does the IBM PC version use the WordStar commands, but version 2.0 also hooks up all the IBM PC keys, such as insert/delete, page up/down, cursor

pad, home/end, and many more.

For the 8087 version, please note that you must temporarily rename the TURBO-87 file to TURBO if you want the TINST program to install your terminal selection with the right runtime program. After running TINST, you can change the updated TURBO file back to "TURBO-87" to distinguish it from the standard TURBO.

Speed

Okay, here's where Turbo Pascal shines, compared with the other Pascals on the market. I wondered how Borland could be so brash in their advertisements claiming Turbo Pascal compiled in seconds while others compiled in minutes. Quite simply, you edit a file by bringing it into RAM then selecting the memory compilation mode, after which Turbo Pascal compiles from RAM just as if you were using a RAM disk—at blazing speed with

no disk accesses.

I had configured my UCSD p-System on my IBM PC to do this, and it sure makes Pascal a fun language to use: I don't even mind the iterations of edit, compile, find the missing semicolon, re-edit, and so on. Turbo Pascal makes this RAM disk-like feature transparent to the user by defaulting to it. Of course, you can also compile to a disk file for execution outside the Turbo Pascal environment (i.e. stand-alone applications or programs).

A simple test of compilation speed was to compile one of the sample IBM programs, ART.PAS, a graphics demo. With my stopwatch in hand, I determined that the 151-line program compiled in 3 seconds! Longer programs such as the MicroCalc demo also compiled in an amazingly short time. The painful part of Pascal—compilation and syntax correction—has been removed. Even novices will appreciate

Q-PRO 4

Q-PRO 4 handles all the micros ... local area networks (with record and file locking), multi-user, single-user, 8-bit, 16-bit, even the new IBM AT.

The user-friendly applications you write with Q-PRO 4 are fully transportable. They run faster. And you can protect them with our author's lock up package.

Q-PRO 4 is the professional developer's package with no limitations. It runs under PC-DOS, MS-DOS, CCP/M, PC Net, NetWare, EtherShare, DNA, CP/M, MP/M, TurboDOS, MmmOST, MUSE, and NSTAR.

And, just in case you don't read reviews or attend seminars, Q-PRO 4 is the one that the computer experts evaluated alongside dBASE II and showed how Q-PRO 4 blows dBASE II away.

	Q-PRO 4	dBASE II	dBASE III
DATA BASE			
#Open files	255	2	10
#Fields	Unlimited	32	128
Record size	Unlimited	1024	1024
Multi key ISAM	Yes	Needs sorting	Needs sorting
LOCAL AREA NETWORKS			
File lock	Yes	No	No
Record lock	Yes	No	No
PORTABILITY			
8-bit → 16-bit	Yes	Yes	No
16-bit → 8-bit	Yes	Yes	No
MISCELLANEOUS			
Formatted data entry	Full	Limited	Limited
Report generator	Full	Limited	Limited
Memory variables	Unlimited	64	256
Programmable function keys	21	0	0

One last word.

If you still write business applications with an old second generation language like BASIC, now's the time to stop ripping yourself off. Q-PRO 4 is the productivity tool that lets you write much better applications in one tenth the time. No exaggeration.

Single-user—\$595; Multi-user—\$795
Demo package available.
Author's lock up available.

And don't dump your dBASE files. A Q-PRO 4 utility will convert them.

Order Q-PRO 4 now.

136 Granite Hill Court
Langhorne, PA 19047
(215) 968-5966 Telex 291-765

quic·n·easi products inc.

that their learning time can be spent more on the language than on compilation syntax checking.

As for execution speed I unfortunately couldn't locate an 8087 to check Turbo Pascal's true speed, but I can compare common benchmarks running on my 4.77 MHz IBM PC and my 8 MHz HP-150. Using the *DDJ* Savage floating-point benchmark that first appeared in November 1983, I produced some new entries for Ray Duncan's floating-point benchmark table that appeared in *DDJ*'s August 1984 issue. This benchmark really tests the round-off errors and significant digits of a language, plus its math speed. Like Turbo Pascal version 1.0, version 2.0 still doesn't include an arctangent function, so I had to do it in math.

As you can see in Table 3 (below), Turbo Pascal version 2.0 hasn't really increased its speed on the IBM PC from version 1.0; the HP-150 excels primarily due to its 8 MHz CPU. While Turbo Pascal doesn't show blazing execution speed, it is very accurate in its math.

The Reference Manuals

My only complaint with the manual is that, in trying to cover the CP/M-80, PCDOS/MSDOS, and CP/M-86 variations in one manual, the summary tables occasionally were incomplete for the 16-bit versions: the tables represented the CP/M-80 implementation correctly but failed to really indicate the additional procedures and functions of the 16-bit implementations. The manual is divided into sections: general information, CP/M-80 specific, MSDOS/PCDOS specific, CP/M-86 specific, and summary tables. In the 16-bit sections covering the MSDOS/PCDOS and CP/M-86 function calls, no syntax definitions were given at all. I really had to study the documentation files on the distribution disk to get the necessary information. This may have been due to an initial uncertainty in their implementation, but this is the second edition of the version 1.0 manual. I hope that Borland will update this manual soon to clean up its errors.

As for the version 2.0 manual, it is short and simple. The descriptions of the overlays, graphics, color, sound, and 8087 enhancements are adequate. Once again, the summary table could be more helpful in indicating which

version 2.0 additions are IBM PC specific, plus be more accurate. Both of the window procedures and the draw procedure were improperly listed. No

mention is made of any differences between the CP/M-80 and 16-bit implementations of overlays and the dispose function, so I trust both work similarly.

Enhancement	IBM PC Only
Reserved word	
overlay	
procedure	
Dispose(var P:Pointer);	
FreeMem(var P:Pointer, I:Integer);	
TextMode(Color:Integer);	XX
TextBackground(Color:Integer);	XX
TextColor(Color:Integer);	XX
GraphColorMode;	XX
GraphMode;	XX
HiRes;	XX
GraphBackground(Color:Integer);	XX
Palette(Color:Integer);	XX
HiResColor(Color:Integer);	XX
Plot(X,Y,Color:Integer);	XX
Draw(X1,Y1,X2,Y2,Color:Integer);	XX
Window(X1,Y1,X2,Y2:Integer);	XX
GraphWindow(X1,Y1,X2,Y2:Integer);	XX
Sound(I:Integer);	XX
NoSound;	XX
function	
MaxAvail:Integer;	
WhereX:Integer;	XX
WhereY:Integer;	XX
pre-defined constants	
4 for text mode selection	XX
16 for color selection	XX
1 for blinking text	XX

Table 2
Turbo Pascal Version 2.0 Enhancements

Computer	MHz	Language	Vers	FPP	Time (sec)	Error
IBM PC(8088)	4.77	Turbo Pascal	2.0	8087	??	??
IBM PC(8088)*	4.77	Turbo Pascal	2.0	--	535	4.6E-3
HP-150(8088)	8	Turbo Pascal	2.0	--	396	4.6E-3
IBM PC(8088)†	4.77	Turbo Pascal	1.0	--	544	5E-3

* Same results with PCDOS and MSDOS versions of Turbo Pascal
† As tested by Jeff Furgal

Table 3
Turbo Pascal Benchmarks
for June 1984 DDJ 16-Bit Toolbox

THE PROGRAMMER'S SHOP™

helps compare, evaluate, find products. Straight answers for serious programmers.

SERVICES

- Programmer's Referral List
- Compare Products
- Help find a Publisher
- Evaluation Literature free
- BULLETIN BOARD - 7 PM to 7 AM 617-826-4086
- Dealer's Inquire
- Newsletter
- Rush Order
- Over 300 products

Free Literature - Compare Products

Evaluate products. Compare competitors. Learn about new alternatives. One free call brings information on just about any programming need. Ask for any "Packet" or "Addon Packet": ☐ ADA, Modula ☐ "AI" ☐ BASIC ☐ "C" ☐ COBOL ☐ Editors ☐ FORTH ☐ FORTRAN ☐ PASCAL ☐ UNIX/PC or ☐ Debuggers, Linkers, etc.

RECENT DISCOVERIES

SCIL - Manage versions, changes to source code, documentation. Minimize confusion, disk space. Interactive. CPM 80, MSDOS \$349

FORTRAN ENVIRONMENT PRICE

MS FORTRAN-86 - Impr.	MSDOS \$350	\$ 225
Intel Fortran-86	IBM PC	NA 1400
DR Fortran-86 - full '77'	8086	500 349
PolyFORTRAN-XREF, Xtract	PCDOS	NA 165

OTHER PRODUCTS

Assembler & Tools - DRI	8086	200 159
Atron Debugger for Lattice	PCDOS	NA 695
CODESMITH-86 - debug	PCDOS	149 139
CURSES by Lattice	PCDOS	NA 125
Disk Mechanic - rebuild	MSDOS	70 65
HS/FORTH - fast	PCDOS	220 210
IQ LISP - full 1000K RAM	PCDOS	175 call
MBP Cobol-86 - fast	8086	750 695
MicroPROLOG	PCDOS	NA 285
Microsoft MASM-86	MSDOS	100 85
MSD Debugger	PCDOS	125 119
MultiLink-Multitasking	PCDOS	295 265
PLIX-86 Debugger	MSDOS	195 169
PL1-86	8086	750 495
PLINK-86 - overlays	8086	350 315
Polylibrarian - thorough	MSDOS	99 95
PolyMAKE	PCDOS	99 95
PROFILER-86 - easier	MSDOS	NA 125
PROFILER - flexible	MSDOS	NA 125
Prolog-86-Learn, Experiment	MSDOS	NA 125
TLC LISP-86-full, liked	MSDOS	NA 235
TRACE86 debugger ASM	MSDOS	125 115
X Shell-IF-THEN-ELSE	MSDOS	295 279

Note: All prices subject to change without notice. Mention this ad. Some prices are specials.

Ask about COD and POs.
All formats available.

0105

"C" LANGUAGE

LIST PRICE	OUR PRICE
MSDOS: C86-8087, reliable	\$395 call
Instant C - Inter., fast, full	NA 500
Lattice 2.1 - improved	500 call
Microsoft C 2.x	500 349
Williams - NEW, debugger	500 call
CPM80' Ecosoft C-now solid, full	250 225
BDS C - solid value	150 125
MACINTOSH: Full, ASM	NA 385

Compare, evaluate, consider other Cs

BASIC

ENVIRONMENT	LIST PRICE	OUR PRICE
Active Trace-debug	86/80	NA 75
BASCOM-86 - MicroSoft	8086	395 279
BASIC Dev't System	PCDOS	79 72
BASICA Compiler - BetterBASIC - 640K	PCDOS	NA 185
CB-86 - DRI	CPM86	600 439
Prof. BASIC Compiler	PCDOS	345 325
MACINTOSH COMPILER with BASICA syntax	MAC	NA 325

Ask about ISAM, other addons for BASIC

FEATURES

LINT-86 - finally a full lint to find subtle bugs - for all MSDOS Cs. \$200.

PROLOG86 Interpreter for MSDOS includes tutorials, reference and good examples. Learn in first few hours. For Prototyping, Natural Language or AI. \$125.

EDITORS Programming

BRIEF - Intuitive, flexible	PCDOS	NA 195
C Screen with source	8080/86	NA 75
FINAL WORD - for manuals	8080/86	300 215
MINCE - like EMACS	CPM, PCDOS	175 149
PMATE - powerful	CPM	195 175
	8086	225 195
VEDIT - full, liked	CPM, PCDOS	150 119
	8086	150 119

UNIX PC

COHERENT - for "C" users	PClike	\$500 475
COHERENT-NCI-Realtime	PClike	695 665
VENIX - "true V7" w/FTN	PClike	800 775
XENIX - "true S3" - rich	PC	1350 1295

Ask about run-times, applications, DOS compatibility, other alternatives. UNIX is a trademark of Bell Labs

LANGUAGE LIBRARIES

C Sharp Realtime-source	MSDOS	NA 600
GRAPHICS: GraphiC-source	MSDOS	NA 195
HALO - fast, full	PCDOS	200 165
Greenleaf for C - full	MSDOS	NA 165
ISAM: C to dBASE-source	8086	150 140
C'Tree-Source, no royalties	ALL	NA 375
dbVista - "Network," Source	MSDOS	495 465
BTRIEVE - many languages	MSDOS	245 215
Index + - no royalties	MSDOS	NA 395
PHACT - with C	MSDOS	NA 250
dBC - by Lattice	MSDOS	NA 250
SCREEN: CView-validate	PCDOS	NA 195
Databurst-C, BASIC	MSDOS	225 215
PANEL-86-many languages	PCDOS	295 265
WINDOWS for C-fast	PCDOS	NA 139

Call for a catalog, literature, and solid value

800-421-8006

THE PROGRAMMER'S SHOP™

128-D Rockland Street, Hanover, MA 02339.

Visa Mass: 800-442-8070 or 617-826-7531 MasterCard

Circle no. 75 on reader service card.

C Helper™

FIRST-AID FOR C PROGRAMS

Save time and frustration when analyzing and manipulating C programs. Use C HELPER's UNIX-like utilities which include:

- DIFF** and **CMP** - for "intelligent" file comparisons.
- XREF** - cross references variables by function and line.
- C Flow Chart** - shows what functions call each other.
- C Beautifier** - make source more regular and readable.
- GREP** - search for sophisticated patterns in text.

There are several other utilities that help with converting from one C compiler to another and with printing programs.

C Helper is written in portable C and includes both full source code and executable files for \$135 for MS-DOS, CPM-80 or CPM-86. Use VISA, Master Card or COD.

Solution Systems™

Call: 617-659-1571

335-D Washington Street
Norwell, MA 02061

ARTIFICIAL INTELLIGENCE

Programming: Learn Fast, Experiment, Prototype

with the nonprocedural language chosen by Japan

PROLOG-86™

implements the "standard" features described in Clocksin and Mellish, has tutorials and sample programs to learn from that include:

- an Expert System
- a Natural Language Interface

1 or 2 pages of PROLOG is often equivalent to 10 or 15 pages in "C" or PASCAL. It is a different way of thinking. Become familiar in one evening, comfortable in days.

CONTEST: "Artificial Intelligence Concepts". All entries that teach key concepts and are clear will get recognition. The best will win \$1,000. Submit by 2/28/85.

AVAILABILITY: PROLOG-86 runs on MSDOS, PCDOS or CPM-86 machines. We provide most formats. The Intro price is \$125.

Solution Systems™

335-D Washington Street
Norwell, MA 02061

Full Refund if not satisfied during first 30 days.
617-659-1571

Compiler Directives

Nothing new here except that the X directive (array index optimization) is CP/M-80 specific and not used in the 16-bit versions. Also, the I directive (I/O error checking on/off) uses the standard Pascal function IoResult, not IoError. Both of these are corrections to David Clark's version 1.0 review.

New Enhancements

Here's a simple overview of what version 2.0 adds. For all versions:

- (1) Overlay system (with code swapping to write programs larger than the memory available for program code)
- (2) Dynamic heap (with a real dispose function to supplement the more restrictive mark and release procedures of version 1.0)

(3) Additional editor commands
For IBM PC and compatibles only:

- (4) Colors
- (5) Graphics
- (6) Windows (graphics and text)
- (7) Sound

For 16-bit versions:

- (8) Optional 8087 support

Overlays

How many times have you composed a program that ended up being too large to compile? Many large application programs are like this. Programmers usually resort to breaking the program into smaller chunks that are independent in function to the others. If you think this sounds like the definition of a procedure or function, you're right, except that it is on a larger scale: groups of procedures and functions are the chunks. Some familiar terms and techniques to achieve this are overlays, chaining, separate compilation, segments, units, external modules, and so on. Let's take a quick look at the most popular techniques to see how Turbo Pascal and its overlays fit in.

Chaining is best exemplified by Microsoft BASIC, whether it is on CP/M, MSDOS, or the IBM PC. Fundamentally, every program has a runtime library of built-in functions plus variable/data space. One program can "chain" to another (i.e., call the other program and relinquish control) with little difference in the programs beyond executing a chain statement. Chained programs also typically like to pass information to one another

without using a disk file—via common variables. Common variables mean that the interpreter or compiler sets aside a specific amount of variable space, protecting it from being destroyed in the chaining process. If the chained programs have equivalent common-variable definition statements, these variables aren't garbage or initialized but retain their values for use by the new program. Key advantages are in passing data; also, with compiled programs, the runtime library need not be loaded again.

Separate compilation and modules let you write parts of the program separately, each with different variables, types, procedures, and functions. After all the separate modules are compiled, a main program (sometimes called a shell program) calls each module as it needs it. This may entail loading all the modules in memory at once or doing memory management and swapping in and out of memory only the modules that are most recently used. Both MS-Pascal and UCSD p-System Pascal implement variations of this. In fact, the UCSD p-System itself is an excellent example of separate compilation; many of its users quickly notice the disk swapping of the numerous parts of its system in and out of memory as needed. Key advantages are that you can write and compile many chunks of a program independently of the others to compose a rather large application.

Overlays are similar to both chaining and separate compilation. As implemented in Turbo Pascal, you may compose a program with as many overlay groups in the source code as you want and compile it all at one time. An overlay group is a sequential list of procedures and functions that can be called independently of the others in the list. Each overlay group sets aside an amount of memory equal to the largest procedure or function in the group, then swaps into this space the procedure or function you call. To use one or more overlay groups in the program, you effectively define which routines don't call each other, then group them together in their own overlay. Key advantages are code swapping for minimal program memory.

In Turbo Pascal, you define an overlay group with a sequential list of routines using "overlay" preceding the

function or procedure declaration. Note that an overlay group ends when a procedure or function definition without the overlay designation is encountered. Hence, you can set up several overlay groups, separating each group with a non-overlay routine—even a dummy procedure of begin/end will do it.

Listing One (page 117) shows a program composed of two overlay groups, each with two procedures. Note that a disk access is performed with each call to a new procedure in an overlay group. As Turbo Pascal compiles your program, each overlay group is compiled to disk under the name of your program with a file type of .000, .001, .002, and so on. This means 1000 overlay groups are possible for a single program! The number of routines per overlay group seems to be unlimited. To actually compile a program with overlays, you must select compilation to the disk; Turbo Pascal can't handle it in its own memory. Likewise, to run the compiled program, you must exit Turbo Pascal and execute the program from DOS—definitely an inconvenience.

The advantages of overlays in Turbo Pascal are in saving program and data space, especially since overlays can be nested. The disadvantages are that you can't compile to memory, the program and overlay files must reside on disk, you can't execute a program while in Turbo Pascal, extra time is needed for the disk I/O for overlay retrieval, extra code is generated for overlay management, and the runtime debugger has problems inside the overlay groups.

I have a greater appreciation of the UCSD p-System separate compilation capability (units and segments) after using Turbo Pascal's overlays. In essence, Turbo Pascal's overlays are the equivalent of UCSD p-System's segments. I have grown fond of the UCSD p-System unit concept, especially for generating a large application program. For smaller jobs, however, the segment/overlay approach is more than adequate.

Dynamic Heap Management

Many languages have the capability to dynamically allocate and deallocate variables and, more importantly, memory space. For example, Micro-

Elegance

Power

Speed



C Users' Group

Supporting All C Users

Box 97
415 Euclid
McPherson, Kansas 67460
(316) 241-1065

Circle no. 17 on reader service card.

VANCE info systems is pleased to announce
THE MOST COMPLETE C FUNCTION LIBRARY AVAILABLE TO DATE!

lib

C FUNCTION LIBRARY

"C-lib" is the most functional library available for software written in C, providing over 200 routines, extending the capabilities of C on the IBM PC. The library is available under the DeSmet (C Ware) C88 compiler, will be available in MicroSoft C, Lattice C and other C compilers, and runs using MS Dos 1.1 and later versions.

Routines are included to handle:

- System date & time I/O
- Date & time math functions
- Extended screen & keyboard I/O
- String & bit-field manipulation
- Degrees/radians conversions
- And contains dozens of functions compatible with Xenix and Unix.
- String & numeric sorting
- Trigonometric & hyperbolic functions
- Asynchronous communications functions

"C-lib" features a unique windowing library transportable with Xenix function calls.

"C-lib" also includes functions to convert floating point numbers from MicroSoft to 8087 NDP format and vice versa.

Documentation is offered in an easy to use printed manual or on disk for your own printing needs, complete with programming examples and follow-up demo programs.

The "C-lib C FUNCTION LIBRARY is offered at only \$145, less than most available today.

For further information on "C-lib" please contact us at:

VANCE info systems

2818 clay street • san francisco, california 94115 • (415) 922-6539

IBM is a trademark of International Business Machines Corp.
C88 is a trademark of Computer Innovations, Inc.
Lattice is a trademark of Lattice Inc.
Xenix, MicroSoft C and MS DOS are trademarks of MicroSoft Inc.
Unix is a trademark of Bell Labs, Inc.
C-lib is a trademark of VANCE info systems.

Circle no. 106 on reader service card.

PROGRAMMER'S DEVELOPMENT TOOLS

IBM Personal Computer Language and Utility Specialists

LANGUAGES:

	List	Ours
8088 Assembler w/Z-80 Translator 2500 AD	100	89
C-86 by Computer Innovations	395	309
CB-86 by DRI	600	429
DeSmet C Compiler with Debugger	159	145
Instant-C by Rational Systems, Interpretive C	500	469
Janus/ADA + Tools by R&R	700	499
Lattice C Compiler	500	299
STSC APL*Plus/PC	595	499
Xenix Development System by SCO	1350	1099

Call for Prices and Information about other Languages.

Special Holiday Season Sale Price!

Mark Williams C Development System \$429

The MWC Development System defines a new standard for C Compilers. The compiler provides excellent benchmarks and extremely fast compilations. In addition, the system includes its own assembler and source level debugger. Best of all, **you can now save an additional \$30 off** our already discounted price. Call for details.

Manufacturer's List Price \$500.

Special Introductory Offer

Save \$50 off manufacturer's list price!
BetterBASIC by Summit Software \$149

This new approach to BASIC programming supports modular structured techniques, large workspaces, incremental compilation, pointers and more. An optional 8087 Math Module is available for \$89.

UTILITIES:

	List	Ours
Btrieve by SoftCraft	245	199
CodeSmith-86 by Visual Age	145	129
Communications Library by Greenleaf	160	139
C-Food Smorgasbord	150	109
C Power Paks from Software Horizons	CALL	CALL
C-Tree by Faircom	395	359
C To Dbase by Computer Innovations	150	139
C-Utilities by Essential Software	149	119
Dr. Halo	100	89
ESP for C or Pascal by Bellesoft	CALL	CALL
FirstTime for C by Spruce Tech	295	269
GraphiC from Scientific Endeavors	195	179
Greenleaf Functions Library	175	139
Halo Color Graphics for Lattice, C1-86	200	125
Panel Screen Design/Editing by Roundhill	295	234
Pfix 86 by Phoenix Software	195	175
Pfix 86+ by Phoenix Software	395	355
Phact by Phact Associates	395	359
Plink-86 Overlay Linkage Editor	395	310
Pmate by Phoenix Software	225	175
Profiler by DWB & Associates	125	99
Too's by Blaise Computing	CALL	CALL
Trace-86 by Morgan Computing	125	115
Windows for C by Creative Solutions	195	159

Prices are subject to change without notice.

**Call for our New Catalog consisting of
200+ Programmer's Development Tools
Exclusively for IBM PC's and Compatibles.**

Account is charged when order is shipped.



1-800-336-1166



Programmer's Connection
281 Martinel Drive
Kent, Ohio 44240
(216) 678-4301 (In Ohio)

"Programmers Serving Programmers"

Circle no. 74 on reader service card.

soft BASIC has its string variable garbage collection. Pascal has two approaches: mark/release and new/discard. Turbo Pascal version 1.0 implemented the mark, release, and new procedures, but no discard procedure. Version 2.0 now fully implements the discard procedure, a nice enhancement.

Why the fuss about discard? Let's look at how different the two Pascal approaches are, then you'll see why. Imagine some sequential stretch of memory where variables are stored. Under the mark/release approach, you specify a "mark" variable as the memory pointer. When you later execute the release procedure with this "mark" variable, it and all variables stored after it are erased. Hence, mark/release defines where new variable space starts by deleting a block of previous variables. New/discard works a bit differently: these procedures use and make available variable space in a random fashion. When you "discard" a variable, the memory space used by that variable can be reused when a new variable is specified by the "new" procedure.

In summary, mark/release specifies a block of RAM for reallocation, and new/discard specifies any variable space for reuse. One last comment: mark/release and new/discard should never be used together—you must select the approach you wish to use.

IBM PC Goodies

Now let's get to the meat of the version 2.0 enhancements: monitor control, graphics, color, sound, and windows for the IBM PC and its compatibles (see Table 2). Borland has been nice to us here; I just hope they can support other computers with these types of enhancements.

Monitor control includes four modes:

- (1) Text mode—25 lines of 40 or 80 characters, color or black and white
- (2) Graph color mode—320 x 200 dots, four color graphics
- (3) Graph mode—320 x 200 dots, black and white graphics
- (4) HiRes mode—640 x 200 dots, black and one color graphics.

In text mode, after selecting the appropriate display format, you can define both the background and the text

colors using the color graphics board. You can select from any of 16 available colors. Whichever monitor you use, the "where cursor" functions give you the current x and y coordinates of the cursor. You can even have the text blink by selecting the text color with a blink offset (add 16).

In the graph color mode, you have control over the background and pen colors. After selecting the background from one of 16 available colors, you define the pen color according to which palette you are using (there are four). For example, palette(0) is composed of background, green, red, and brown. This will be the most used graphics mode.

Graph mode is similar to graph color mode, except that background color is black and pen color is white. If you are using an RGB monitor, this mode is enhanced: you can select a background color from the 16 available and a pen color from one of two palettes.

In HiRes mode, you select only the pen color; the background is always black, independently of whether your monitor is color or black and white. The pen color is selectable from the 16 available.

Now, how do you use graphics? The plot procedure basically plots a point on the screen. The draw procedure plots a line between two points. In both cases, you have control of the pen color. The x and y coordinates for each point are defined with the origin (0,0) in the upper lefthand corner of your monitor; the lower righthand corner becomes either (319,199) or (639,199), depending upon the resolution used.

This orientation is the same as IBM BASIC, but different from my UCSD p-System Turtlegraphics where resolution control is the same, but the lower lefthand corner is (0,0) and the upper-righthand corner is (319,199) or (639,199). I feel more comfortable with the latter, but most of us can use either approach.

The area fills, circles, etc., of IBM BASIC are missing here, but these are not a great loss. One feature that I do miss from my Turtlegraphics is the ability to rescale the screen to use any coordinate system I wish. With a single statement, I could set the x-axis of a graph to be the years 1900 to 2000 and

Turbo Pascal, Version 2.0

**Company: Borland International,
4807 Scotts Valley Drive,
Scotts Valley, CA 95066**

**Computer: CP/M-80, or CP/M-86,
CCP/M-86**

Price: \$49.95

Reviewed by David D. Clark

Before my original review of Turbo Pascal, version 1.0, made it into print, Borland announced a new release. Because the original was so good, I was eager to get my hands on the new one. Again I have not been disappointed. The revised product has several improvements, most of which will have the biggest impact on users of the IBM PC and compatibles. That version is reviewed by Kachigan starting page 107, but the editors of *Dr. Dobbs's* let me look over the new 8-bit version as well.

The editor has been expanded slightly with some additional commands, still similar to WordStar's. The standard procedure, Discard, has been implemented. An additional new function, MaxAvail, is provided that returns the amount of memory available for allocation of dynamic storage. The FreeMem procedure is now provided to release blocks of dynamically allocated memory from the heap. It is symmetrical to the GetMem procedure previously provided.

The big news for the 8-bit version is the inclusion of dynamic overlay facilities. Truly huge programs can be built with this facility. It allows you to specify procedures or functions that can be read into memory when required. It works this way: In the source file, the new reserved word "overlay" is placed before the procedure or function keywords of those subroutines that you wish to run as overlays. Groups of subroutines declared consecutively in the program source will be placed in the same overlay file after compilation. At runtime, routines from the same file will occupy the same space in memory and will be retrieved from disk as they are invoked. To place overlays in different overlay files, the routines must be separated by the declaration of a nonoverlay object. A dummy type declaration will do. Nested overlays are allowed to any depth.

This overlay strategy places some restrictions on their use. Since overlays

in the same file run at the same location in memory, they may not call one another, although overlays placed in a separate file, running at a different memory location, may. Also, since memory space for overlays is allocated statically during compilation, the space in memory that they would occupy is not available for anything else; for example, the heap does not increase in size when no overlays are executing. Finally, programs with overlays must be compiled to disk. You will cause compiler error if you try to compile such a program in memory.

The space requirement for this new flexibility is quite modest; it now requires 28K of memory to run Turbo on an 8-bit system instead of 27K. The 16-bit version, with windowing and all that other stuff, now takes 33K. In either case, it is still an incredibly small system.

The first time I reviewed this product, the only thing I could find to complain about was the documentation. It wasn't bad, just a little rough. It's improving, it's improving. Apparently some of the errors in the programming examples were corrected before version 2 was announced. Some other revisions have been made to the manual as well. Some sections still grate on my ears, and some words are still broken at odd places, but the basically good manual has gotten better.

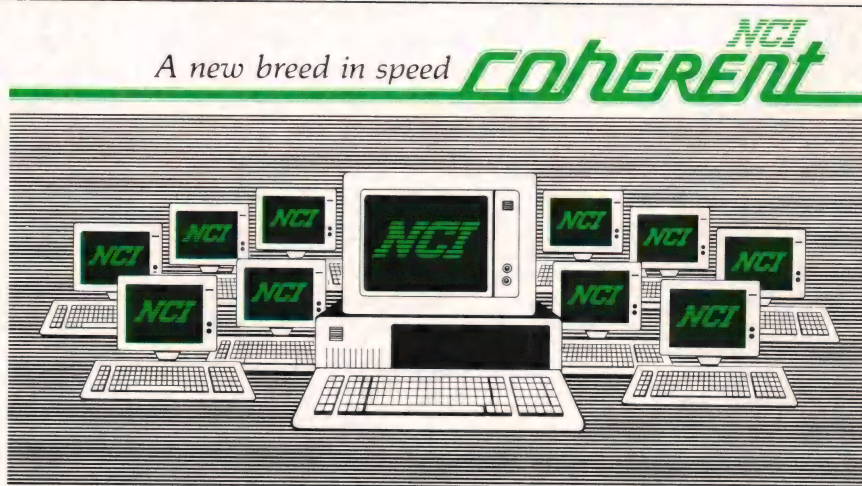
Because of the improvements to the system, I discovered a shortcoming that I hadn't noticed in the original release: there is no Exit procedure. Let me explain. Because of the new overlay capability, I transferred a big, nasty, variable step size, variable order, variable method numerical integration program written in UCSD Pascal to CP/M for use with Turbo. That program was originally written in fairly unstructured Fortran. When the program runs, it is often necessary to return from a deeply nested group of procedure calls without "unwinding" back through them all. Standard Pascal can do this with a goto statement to any location in a program. UCSD Pascal allows gotos only within a block, just like Turbo Pascal. As an alternative, UCSD provides a standard procedure called Exit, which will exit from the subroutine named as its single argument. It is used when unwinding from a deeply recursive set of calls

would be clumsy. Turbo does not supply such a mechanism.

The global goto of standard Pascal is difficult to implement on a one-pass compiler such as the Turbo and UCSD products. An Exit procedure is fairly simple, though. It just involves traversing the chain of activation records until it passes the one sought. If anyone at Borland is listening, an Exit procedure would sure be nice.

In summary, I found version 2.0 of

Turbo Pascal to be incrementally better than its predecessor. The documentation is being polished up. The editor has been expanded slightly. A couple of useful new procedures have been included. The overlay facilities add a lot of flexibility to an already excellent product. Maybe most important, these enhancements have not made the system large and unwieldy. Turbo Pascal is still small, fast, a pleasure to use, and only \$49.95.



The only UNIX compatible operating system that supports 11 simultaneous users on 1 IBM PC XT

Only NCI COHERENT offers powerful multiuser capability with UNIX compatibility.

NCI made COHERENT a multipurpose operating system. We extended it to suit a wide variety of applications. For example, it can now be used to turn a PC into a data multiplexer or an inexpensive database machine in a local area network. NCI COHERENT can also be used for high speed protocol conversion and process control. We've even written a real-time version of COHERENT, exclusive to NCI, which will run on an intelligent peripheral board.

NCI has enhanced the COHERENT kernel making it much faster. Then we added commands that make programming easier, including

a source code control system, plus many UNIX System V features.

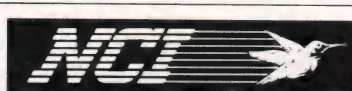
We also gave it exclusive support for a wide variety of hardware and PC compatibles. NCI COHERENT comes with full technical support and complete documentation, organized in the UNIX manner.

A variety of licensing options are available including runtimes, production runtimes, and driver kits.

Remember, when you license COHERENT your license includes multiuser and our low license fees

make your application far more economically feasible.

The NCI technical design team that engineered the improvements to COHERENT is also available on a contract basis.



Performance engineered into every product.

NETWORK CONSULTING INC.
Suite 110, 3700 Gilmore Way,
Burnaby, B.C. Canada V5G 4M1
Phone: (604) 430-3466

*UNIX is a trademark of Bell Laboratories. COHERENT is a trademark of Mark Williams Co. IBM PC and IBM PC XT are trademarks of International Business Machines Corp.

Circle no. 67 on reader service card.

the y-axis to be dollars from \$1,000 to \$10,000. To do this in Turbo Pascal, you have to define a special procedure to scale every x and y coordinate pair before you plot or draw. Also remember that, because the x and y coordinates must be integers, you must truncate whatever your scaled values are back to integers or Turbo Pascal will not accept them.

The first thing I tried after plotting the trace was to label my graph. You do this by overlaying text and graphics—with a few limitations. Characters can be placed only in the grid of 24 lines by 40 characters using the GotoXY procedure, and the text color will be whatever the current pen color is. My Turtlegraphics lets me put a character or string at any point in the graphics screen. Graph labeling could be nicer, but at least Turbo Pascal has the same capability as IBM BASIC.

Plotting speed is comparable to both IBM BASIC and UCSD p-System Turtlegraphics—not slow. Listing Two (page 119) illustrates a simple graphics program: a sine wave with axes and

a title for the plot. It took about 34 seconds to plot everything, most of which was math computation time. When I removed the x and y computations from the for/next loop and just plotted a fixed point, the program took less than 2 seconds!

Some final comments: Selecting any of the graphics modes will clear the screen; if you wish to clear the graphics screen, you must reselect the current graphics mode (the ClrScr procedure works only in text mode). If you have both monochrome and color monitors, you must work on only one of them—you cannot control both simultaneously. And, finally, there is no way to get a hard copy of the current graphics screen. If I could add a few things, I'd like to be able to rescale the x and y axes and to have a graphics dump to my printer. These would make Turbo Pascal a superb graphics tool.

Windows! We do windows! Yes, Turbo Pascal has windows for text and windows for graphics. In graphics, GraphWindow defines what part of the screen you can plot on. If you draw

a line that crosses through the window, only the pixels within the window are affected; everything outside it remains unchanged. In text mode, Window defines where the upper left screen coordinates (1,1) are. All console output appears only within the window, long lines wrap around the window, and the GotoXY function even works.

Sound or NoSound now beep, play songs or make a siren (this is actually part of the SOUND demo program). Quite simply, you specify the tone frequency in Hertz with the Sound procedure, vary the time between tones with the Delay procedure, and turn off the sound with the NoSound procedure.

Conclusions

I am well pleased with Turbo Pascal. It sure comes close in many areas to my UCSD p-System Pascal. However, many of the standard Pascal features missing in version 1.0 are still missing here. They are:

- (1) The Get and Put procedures for I/O and the file buffer variables do not exist. You must use the extended capabilities of Read, Readln, Write, and Writeln to handle all I/O. When translating from other Pascals, this will be a large stumbling block.
- (2) The Goto statement is restricted to transferring control only within the current block, not outside of procedures or functions.
- (3) The Page procedure is not implemented.
- (4) Variable packing and unpacking are not within your control; Turbo Pascal decides itself. Hence, the reserved word Packed has no effect, and the procedures Pack and Unpack are not implemented.
- (5) Procedures and functions cannot be passed as parameters to other subroutines.

Although every Pascal attempts to follow Jensen & Wirth as a guideline to standard Pascal, very few can restrain themselves from deviating on some functions and adding embellishments. Turbo Pascal adds its share of niceties to Pascal: The UpCase function and screen commands (ClrScr, ClrEol, etc.) are a real help. Also, a simple access to CP/M and MSDOS/PCDOS function calls is provided via a procedure and register record.

I'd recommend Turbo Pascal to any-



TOTAL CONTROL WITH LMI FORTH

PC FORTH™
 IBM PC & XT,
 HP-150,
 Macintosh,
 Apple II,
 CompuPro,
 Sage & CP/M-68K,
 Wang PC,
 All CP/M and
 MSDOS computers.

Try the professional language offering the utmost performance in the shortest development time. Transport your applications between any of our enhanced 83-Standard compilers or expanded 32-bit versions. Choose from our wide selection of programming tools including native code compilers, cross-compilers, math coprocessor support, and B-Tree file managers. All fully supported with hotline, updates, and newsletters.

Laboratory Microsystems Incorporated
 Post Office Box 10430, Marina del Rey, CA 90295
 Phone credit card orders to (213) 306-7412

LMI

Circle no. 55 on reader service card.

one interested in Pascal, whether novice or expert. When I talked with one of Borland's technical wizards, he clued me in to a version 3.0 that is coming in the next six months. A major planned addition is complete separate compilation capability. If that happens, I'll probably put away my UCSD p-System Pascal for good. It still shines in better screen control (via screen_ops), communications support (rem_unit), more enhanced graphics (Turtlegraphics), and separate compilation (units), but I'm sure the folks at Borland can come close.

While I'm discussing what's missing, let me give other UCSD p-System Pascal

people a quick review of what needs to be translated from your UCSD p-System programs to make them Turbo Pascal programs. Not implemented:

- (1) units
- (2) exit
- (3) pwriteln(*)

Modifiable:

- (1) io_result—different error number
- (2) segments—overlays
- (3) screen_ops—various screen functions
- (4) close—different syntax
- (5) reset—must use assign and reset, different syntax
- (6) rewrite—must use assign and rewrite, different syntax

(7) turtlegraphics—quite a bit restricted

(8) strings—don't default to 80 characters long, there is no default length at all

(9) underscore—Turbo Pascal doesn't ignore them, they become significant characters in names.

DDJ

Software Reviews (Text begins on page 106)

Listing One

```
program over;
{this demonstrates Turbo Pascal's overlay features.
  File:   over.pas, over.com

  there are two overlay groups in this program
  Files:  over.000 --- proc1a, proc1b
          over.001 --- proc2a, proc2b  }

type
  string80 = string[80];

var
  s: string80;

overlay procedure proc1a;
begin
  writeln('This is the first procedure in overlay #1');
  readln(s);
end;

overlay procedure proc1b;
begin
  writeln('This is the second procedure in overlay #1');
  readln(s);
end;

procedure dummy;
{this forces the two separate overlay groups}
begin
end;

overlay procedure proc2a;
begin
  writeln('This is the first procedure in overlay #2');
```

(Continued on next page)

**IBM-PC
SYMBOLIC
DEBUGGER**

New Release 1.8 — **SOLID GOLD**

CodeSmith™-86

B:\fabcode.COM
CodeSmith-86

Also runs on some IBM-PC Compatibles

AX=8086	SP=8087	SS=1983
BX=0000	BP=0000	DS=1984
CX=0000	SI=0000	ES=1985
DX=1138	DI=0000	CS=2001
		IP=0001

	PL	ZR	NC	NV	UP	NA	PE	EI	
2001:0000	53			24	IO_INIT:	PUSH	BX		;TAG A LINE
2001:0001	9BDEC2					FADDP	ST(2),ST		
2001:0004	BB3100					MOV	BX,Offset VECTOR_TABLE_2		
2001:0007	803E5E-			34		CMP	DOS_VERSION_NUM,'2'		;BREAKPOINT SET
2001:000C	7305					JAE	TRASH_IT		
2001:000E	BB0100					MOV	BX,Offset VECTOR_TABLE_1		
2001:0011	EB02					JMP	Short LONG_LABELS_ARE_OK_AS_YOU_LIKE		
2001:0013	F2AB	00777			TRASH_IT:	REPZ	STOSW		;STOP 777th TIME
2001:0015					LONG LABELS_ARE_OK_AS_YOU_LIKE:				
2001:0015	8DAD63-					LEA	BP,WIERD_CODE + 2[DI]		
2001:0019	240C					AND	AL,00011100B		;CHANGE RADIX
2001:001B	45					DB	69		

MEMORY DUMP

>>DOS_VERSION_NUM
Absolute Address=03C9E
Segment Offset=03C4:005E

1984:0050	41	53	43	49	49	20	53	55-50	50	4F	52	54	20	32	20	ASCII SUPPORT 2
1984:0060	20	20	20	20	43	6F	64	65-53	60	69	74	68	20	38	36	-- CodeSmith-86
1984:0070	20	40	41	4B	45	53	20	44-45	42	55	47	47	49	4E	47	MAKES DEBUGGING
1984:0080	20	41	20	42	4C	41	53	54-21	20	20	20	20	20	20	20	A BLAST!

WNDW 2 LVL 1

It's here—THE Multi-Window Interactive Debugger that's STATE-OF-THE-ART.

- Scroll Up/Down thru full-screen disassemblies & memory dumps
- **Load and Write** Commands much easier, more powerful than DEBUG's
- "Snapshot" a complete debugging state onto disk—resume later
- True passpoints and execution path counters
- SCREENSAVE mode saves and restores user's graphic display when breakpoint hit
- Disassemble selected ranges of memory code to disk—compatible with IBM Assembler
- Stop on data Read/Write or memory range access

Hot-Line technical support

The Professional's Choice—CodeSmith-86

Multiple copies purchased by:
Lotus Development Corp., MicroPro, VisiCorp, IBM.

Requires MS-DOS & 160K RAM.
OEM and dealer inquiries invited.

VISUAL AGE

642 N. Larchmont Blvd. • Los Angeles, CA 90004

Circle no. 108 on reader service card
CodeSmith, TM International Arrangements, Inc.
MS, TM Microsoft Corp.
IBM, TM International Business Machines Corp.

Send **VISUAL AGE** • 642 N. Larchmont Blvd. L.A., CA 90004 • (213) 439-2414
Name **CodeSmith-86** @ \$145 + \$4 shipping (each) • California residents add 6.5% sales tax.

Company _____
Address _____
City _____ State _____ Zip _____
Acct. No. _____
Exp. Date _____
☐ VISA ☐ MasterCard
☐ Payment Enclosed ☐ COD

CALL (213) 439-2414
\$145.00


```

    readln(s);
end;

overlay procedure proc2b;
begin
    writeln('This is the second procedure in overlay #2');
    readln(s);
end;

begin
    clrscr;
    writeln('Disk access for overlay group #1');
    proc1a;
    writeln('Disk access for overlay group #2');
    proc2a;
    writeln('Disk access for overlay group #1');
    proc1b;
    writeln('Disk access for overlay group #2');
    proc2b;
    writeln('Done');
end.

```

End Listing One

Listing Two

```

program graph;
{demonstrates Turbo Pascal graphics}

var
    s: string[80];
    pi180: real;
    i,x,y: integer;

begin
    graphcolormode;      {set 320 x 200 color graphics}
    graphbackground(0);  {set background to black}
    palette(0);          {colors are black/green/red/brown}

    draw(0,0,0,199,1);
    draw(0,199,319,199,1);
    draw(319,199,319,0,1);
    draw(319,0,0,0,1);   {draw a frame}

    draw(0,100,319,100,1); {draw center line}
    gotoxy(6,2);
    writeln('This is a Turbo Pascal Graph');

    pi180:=3.14/180;
    for i:=0 to 720 do
        begin
            x:=trunc(i/2);
            y:=trunc(100+75*sin(i*pi180));
            plot(x,y,2);   {draw a sine wave}
        end;

end.

```

End Listings

NOW C HERE! CROSS SOFTWARE for the NS32000

Also Available for IBM PC

INCLUDES:

- * Cross Assembler *
- * Cross Linker *
- * Debugger *
- * N.S. ISE Support *
- * Librarian *
- * Pascal Cross Compiler *
- * C Cross Compiler *

U.S. prices start at \$500

SOLUTIONWARE

1283 Mt. View-Alviso Rd.

Suite B

Sunnyvale, Calif. 94089

408/745-7818 * TLX 4994264

Circle no. 96 on reader service card.

A general purpose programming language for string and list processing and all forms of non-numerical computation.

SNOBOL4+ —the entire

SNOBOL4 language with its superb pattern-matching facilities • Strings over 32,000 bytes in length • Integer and floating point using 8087 or supplied emulator • ASCII, binary, sequential, and random-access I/O • Assembly Language interface • Compile new code during program execution • Create SAVE files • Program and data space up to 300K bytes RAM

Have you tried SNOBOL4+?

With **ELIZA** & over 100 sample programs and functions

For all 8086, 88 PC/MS-DOS or CP/M-86 systems, 128K minimum 5 1/4" DSDD, specify DOS/CPM format

Send check, VISA, M. C. to: **\$95**
Catspaw, Inc. plus '3 s/h
 P.O. Box 1123 • Salida, CO 81201 • 303-539-3884

Circle no. 20 on reader service card.

by Michael Wiesenberg

Bob Levin, hotshot young programmer at I-Q Industries in the heart of Silicon Valley, stands at one of I-Q's permanent coffee stations and stares forlornly at the empty coffee pot. The burner has been on all night and all that remains in the pot is a dark brown paste.

Grey Scrivener, senior technical writer, glides around the corner. He smiles as much at Levin's predicament as at his tattered tennis shoes, wrinkled wash pants, and T_EX t-shirt ("!You can't do that in horizontal mode"). Scrivener smooths an invisible wrinkle in his Calvins and takes over. He carries the pot into the men's room and washes out the muck. When he returns, Levin has been joined by Spotswood Gilbert, senior programmer, who holds a raisin Danish in one hand and a chocolate brownie in the other.

Marian Smith, system operator, arrives, trailing a cloud of smoke, which Scrivener fans away with his hands. She sets her cigarette on the edge of the table. Burn marks on the table indicate that she has probably done this before. "Baby Huey's down. You been messing with the operating system again, Bobby?"

"The changes I make only speed the system up. If it's down again, you can blame all the unsupported utilities. You'd think a company as big as this would get a full-screen editor for program development."

Sally McRae walks up, having overheard the last part of the conversation. She has on Nikes, baggy shorts over purple tights, and a Cocolat t-shirt. "Maybe you didn't crash the system, but you sure brought it to its knees. What're you doing to use up all that cpu time?"

Levin, uncomfortable around attractive women, is too embarrassed to look McRae in the eyes, so he settles on her nose. "I'm trying to solve a puzzle."

Gilbert takes the last bite of his Danish and starts on the brownie. "Is it as tough as that last one?"

"Nah. This was a contest by a puzzle magazine. Winner got a new IQ2557Q portable computer."

Smith notices the veneer top of the coffee station table starting to char and shoves the cigarette back in her mouth. "Hey, I'd like one of those. Even with the employee discount, they're not cheap."

"Well, the funny thing is I don't think the winner used a computer to come up with the winning answer. I think a computer could have found a better solution."

Scrivener adjusts his Stanford ring. "You mean this puzzle had more than one answer?"

"No, but I'm pretty sure there's a better answer than the one that won the contest. I just don't think any of the entrants found it because I don't think anybody knew how to write a program to solve the problem. I wish I'd heard about it before the contest was over. I have a program running right now that should give me the answer that would have won. Listen:

"Assign a numerical value to each letter of the alphabet, starting with 1 for A and going up to 26 for Z. Any word in the English language has a value obtained by multiplying the values for each of its letters. For example, the word *hello* is worth 86,400, obtained by multiplying $8 \times 5 \times 12 \times 12 \times 15$. Which English word is equal to exactly 1,000,000? If there is none, which is closest? Only words found in *The Random House Dictionary of the English Language* (unabridged edition) can be used. No capitalized words, none with hyphens or other embedded punctuation, nor those designated as foreign."

McRae laughs. "How about a program that does my work for me at noon

while I drive to Gelato Classico?"

Gilbert finishes the brownie and pours the last of the coffee. "Hmm. I can see right away that the word has to have at least five letters."

Scrivener dumps the old filter and grounds, measures fresh coffee into a new filter, replaces the holder, and presses the BREW button. "I see that. The four-letter combination with the highest value would be zzzz, and that multiplies out to considerably less than a million."

Gilbert presses buttons on his watch. "How about *linger*? That's 952,560."

Levin pushes the straggly blond hair off his forehead. "That's 47,440 off, and you're just guessing. I can give you a better guess: *single*. That's 1,005,480, only 5,480 off. But there's a scientific way to figure it, and I'm sure it can be done only by computer."

* * * * *

Well, how about it, folks? Can you devise a program that finds the right word? And, having done that, can you tell us what that word is? Your program must be short and elegant. The algorithms can be demonstrated in a good pseudolanguage if you wish or perhaps in flowcharts. The best solution wins a t-shirt and will be published here.

DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 199.



The Tools You Need To C You Thru.

Now the *WizardWare™*
Applications Programmers Toolkit
provides everything you need
to increase your C programming productivity.

APT™ features:

- File Handlers
 - Direct Access
 - Keyed Access
- Generic Terminal Driver
- String Handling
 - Manual On Disk (45 pages)
 - Tutorial On Disk (33 pages)
 - Detailed Brochure on request
- String Math
- Screen Generator
- Report Generator
- FIFO Queue Routines
- Source Code

UNIX™ System V Version (available 1-1-85) . \$995

MS-DOS™ Version \$495

BDS C™ Version \$395

Manual Only* \$50

*Manual Cost will be applied to APT™ purchase price within
30 days (\$10 re-stocking charge). U.S. funds only, please.

Trademark owners: UNIX™ (AT&T Bell Labs), MS-DOS™ (Microsoft, Inc.),
BDS C™ (BD Software), WizardWare™ and APT™
(Shaw American Technologies)

Call: (800) 443-0100 Ext. 282

Ask for APT™ or Send Check To:

Shaw ☆ American Technologies

WizardWare™

830 South Second Street · Box 648

Louisville, KY 40201, U.S.A.

(C.O.D. and Foreign Orders - Add \$5 Shipping/Handling)

References: Bank of Louisville, Citizens Fidelity Bank, Louisville Chamber of Commerce



Circle no. 86 on reader service card.

WIZARD C

Fast compiles, fast code and great diagnostics
make Wizard C unbeatable on MSDOS. Discover
the powers of Wizard C:

- ALL UNIX SYSTEM III LANGUAGE FEATURES.
- UP TO A MEGABYTE OF CODE OR DATA.
- SUPPORT FOR 8087 AND 80186.
- FULL LIBRARY SOURCE CODE, OVER 200 FUNCTIONS.
- CROSS-FILE CHECKS OF PARAMETER PASSING.
- USES MSDOS LINK OR PLINK-86.
- CAN CALL OR BE CALLED BY PASCAL ROUTINES.
- IN-LINE ASSEMBLY LANGUAGE.
- 240 PAGE MANUAL WITH INDEX.
- NO LICENSE FEE FOR COMPILED PROGRAMS.

The new standard for C Compilers on MSDOS!

Only \$450

WSS

For more information call (617) 641-2379

Wizard Systems Software, Inc.

11 Willow Ct., Arlington, MA 02174

Visa/Mastercard accepted

Circle no. 116 on reader service card.

ADVERTISERS!



Help Us Celebrate 100 Issues of
Dr. Dobb's Journal in our

FEBRUARY GALA ANNIVERSARY ISSUE

Space reservation deadline December 6, 1984

Materials deadline December 14, 1984

Contact

Walter Andrzejewski
Shawn Horst
(415) 424-0600

Beth Dudas
(S. West and S. California)
(714) 643-9439

Dr. Dobb's Journal 2464 Embarcadero Way Palo Alto, CA 94303



Circle no. 132 on reader service card.

INTRODUCING Interface Technologies' Modula-2 Software Development System

The computer press is hailing Modula-2 as "the next standard in programming languages." Modula-2 combines the strengths of its popular predecessor—Pascal—with the features that made the C language appealing, like independent compilation and direct hardware control.

But until today, no company offered a Modula-2 system that made software development fast, easy and efficient.

The fast, powerful tool for programmers

Now that breakthrough is here: Interface Technologies' Modula-2 Software Development System for the IBM® PC, XT, AT and compatible computers gives programmers the same quantum leap in productivity that spreadsheets and word processors gave to end-users. It can reduce monotonous wait time, dramatically increase speed, help eliminate thoughtless mistakes, and free you to become more creative in all your programming efforts.

How to speed input and eliminate 30% of errors

Thirty percent of programming mistakes are syntax errors and simple typos in the program structure. Our "syntax-directed" Modula-2 editor does away with these time-consuming headaches forever.



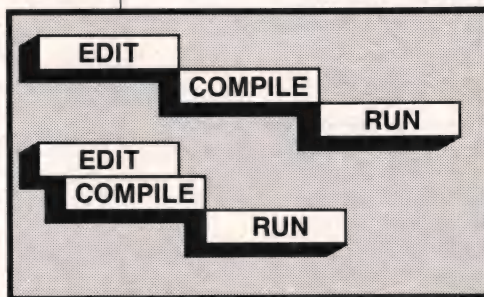
Enter complete statements with one keystroke.

It also speeds input by reducing manual typing as much as 90%, letting you enter statements with a single keystroke. For example, if you type a capital "I" at the beginning of a line, the editor completes the logical "IF THEN" statement automatically, so you can concentrate on what you want to program, rather than your typing.

The editor locks out errors, finishing each statement and procedure in perfect accord with the standardized rules of Modula-2. It also indents and formats your text automatically, making programs easy to read and maintain, an important feature on big projects.

And if you leave an undefined variable or data type, the editor detects the mistake and gives you the option of on-line "help" to correct it. No other programming text editor offers you this much innovation.

How to turn "wait time" into "work time"



The Interface Technologies Modula-2 Software Development System saves time by compiling while you edit.

Most of a programmer's time is spent waiting, and the biggest culprit is usually the compiler. Our compiler

THE ANATOMY OF A

turns this wait time into work time, with a technical innovation we call "background" compilation.

With background compilation, every moment you spend writing or editing a Modula-2 program, it's automatically being compiled into object code, line by line as you work!

When you're finished editing, all that's left for the compiler to do is a quick mopping up that generates optimized native code in a single pass.

How quick is "quick"?

Thanks to background compilation and the fact that the compiler itself is so fast, Interface Technologies' compiler can turn 100 lines of typical Modula-2 program text into optimized machine code in less than five seconds.

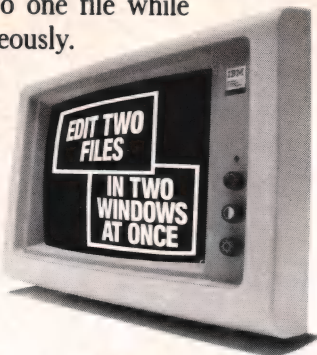
And the Interface Technologies Modula-2 Software Development System compiler produces compact code that has execution speed superior to that produced by any other Modula-2 compiler presently available to individuals and firms involved in software development.

How to do two things at once

Along with the syntax-directed editor and background compilation, Interface Technologies' Software

Development System gives your screen multiple windows so you can refer to one file while you edit another—simultaneously.

Concurrent editing of multiple files is particularly useful when you're doing programming work intended for separate compilation, and Interface Technologies has the only Modula-2 development system on the market that provides you this helpful benefit.



Work with multiple files faster, easier in windows.

How preprogrammed modules speed development

One of the advantages of Modula-2 is that it lets you build large, reliable programs quickly, by linking smaller "building-block" modules.

The development system's toolkit of precompiled program modules includes the standard Modula-2 library, and adds exclusive link-and-run modules for color graphics support, sound, and direct calls to the

You can use it on any IBM® PC, XT, AT or compatible with two DSDD floppy drives and 320K RAM.

You get a thoroughly indexed, comprehensive user's manual and free telephone support from Interface Technologies.

But the most important thing you get is the future, and the programming language of the future is Modula-2.

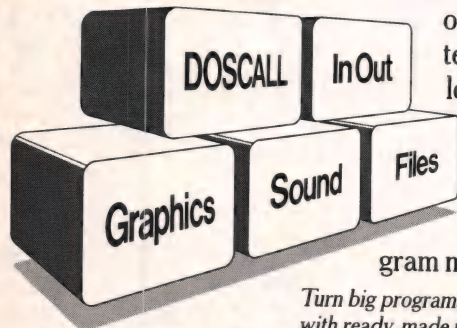
For more information, or to order the Modula-2 Software Development System, call 1-800-922-9049 today. In Texas, call (713) 523-8422.

To order by mail, or to request further information, fill out and mail the coupon below.



IBM is a registered trademark of International Business Machines Corporation.

BREAKTHROUGH



operating system. Plus you get low-cost updates from Interface Technologies' growing library of program modules.

Turn big programs into smaller projects with ready-made modules.

Increase productivity for \$249

Interface Technologies' Software Development System is fast, powerful and unlimited. It works so well that it's the same tool Interface Technologies is using to write business and consumer applications in Modula-2.

For \$249, you get the syntax-directed editor and compiler, linker, module library and tutorial that will have even modestly experienced programmers writing in Modula-2 in days. And you have full rights to your work; there's no license fee for programs you develop with our system.

NAME _____ PHONE _____

COMPANY or SCHOOL _____

MAILING ADDRESS _____

CITY _____ STATE _____ ZIP _____

PLEASE SEND ME _____ copies @ \$249 each.

MAIL REQUESTS TO:
INTERFACE TECHNOLOGIES CORPORATION
3336 Richmond, Houston, Texas 77098

ITC will accept personal checks, money orders or the following credit cards:
Visa/American Express/MasterCard

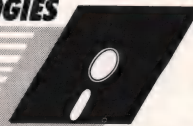
Credit Card # _____ Exp. Date _____

Interbank number {MasterCard}: _____

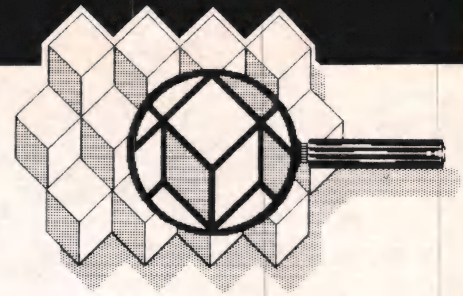
Your Signature _____

Texas residents, add 6.125% Sales Tax.

INTERFACE TECHNOLOGIES



**MODULA-2 SOFTWARE
DEVELOPMENT SYSTEM**



by R. P. Sutherland

Unix Directories

The Unix market is growing at a frantic rate. The number of licenses for Unix and Unix imitations is a quarter of a million and that figure is projected to quintuple over the next two years! Two Unix software directories containing 400 sources each can put folks in touch with suppliers of Unix applications software. Onager Publishing has announced the second edition of *Unix Applications Software Directory*. Cost: \$50.00. Onager Publishing is at 6451 Standridge Ct., San Jose, CA 95123 (408) 225-3541. Another resource is the *Unix System Encyclopedia*. In addition to alphabetical and categorical listings of sources and programs, a third of the book includes articles that describe hardware systems and industry trends. *The Unix System Encyclopedia* is priced at \$34.95, available from Yates Ventures, 4962 El Camino Real, Suite 111, Los Altos, CA 94022 (415) 964-0130.

Unix Utilities

A \$90 **Modula-2** system for the Fortune 68000 system running Unix is available from Modula-2 Corporation. The system allows 128K of workspace for Modula-2 programs, which may be divided between code and data in any ratio. The compiler and interpreter are supplied with the basic I/O modules described by Wirth in *Programming in Modula-2*. Modula Corporation is at 1673 West 820 North, Provo, UT 84601 (800) LILITH2. **Reader Service No. 101.**

A **Writer's Workbench** for Unix System V and 4.1 and 4.2 Unix systems is available from International Data Services. *Writer's Workbench* consists of 25 computer programs that will do such things as proofread text,

analyze spelling and punctuation, and check for sentence length, structure, and voice. *Writer's Workbench* provides on-line information about English usage and allows users to establish their own standards for text analysis. *Writer's Workbench* is distributed at a cost of \$2,000.00. Call International Data Services at (408) 986-1972. **Reader Service No. 103.**

C Tools

Complete Software, Inc., and Catalytic Corporation, both of Cambridge, Massachusetts, have announced some interesting C tools. Complete Software has introduced a **menu-driven C language debugger** that operates independently of host systems. CDEBUG is an interactive tool that symbolically debugs C source code routines on any software or hardware that runs C. Priced from \$300.00, CDEBUG is available from Complete Software, Inc., 60 Aberdeen Ave., Cambridge, MA 02138 (617) 492-5305. **Reader Service No. 105.** Catalytic Corporation has a checkout compiler for the C programming language as well as a C interpreter. The **Safe C Compiler** adds runtime checking to C programs. The **Safe C Interpreter** provides interactive execution of C programs. Catalytic Corporation is at 55 Wheeler St., Cambridge, MA 02138 (617) 497-2160. **Reader Service No. 107.**

C Compilers for Macintosh and Lisa are available from Softworks Limited and Consulair Corp. Softworks Limited has a triple pass compiler system. The C library includes system interface functions, Unix functions, and complete interface to all Macintosh ROM routines. The Lisa implementation will produce programs for either Macintosh or Lisa. The package is \$395.00 for Macintosh and \$695.00

for Lisa. Contact Softworks Limited, 607 W. Wellington, Chicago, IL 60657 (312) 975-4030. **Reader Service No. 109.** Consulair Corp. is shipping a C compiler and support library for the Macintosh (or Lisa under MacWorks). Use of Mac C and the Mac C Toolkit requires Apple's Macintosh 68000 Development System (assembler/debugger). Mac C is \$295.00 and Mac C Toolkit is \$175.00. Consulair Corp. is located at 140 Campo Drive, Portola Valley, CA 94025 (415) 851-3849. **Reader Service No. 111.**

Apple Stuff

A cross assembler that allows development of MC68000 assembler programs on Apple II computers has been made available for \$100.00. Allen Systems' **SX-68 cross assembler** is written in 6502 assembler. Complete access to DOS 3.3 as well as the instruction set specified by Motorola for the MC68000 are supported. Allen Systems, 2151 Fairfax Road, Columbus, OH 43221 (614) 488-7122. **Reader Service No. 113.**

DOS 4.0 for the Apple II family has been released by Rune Software for \$95.00. The company claims that DOS 4.0 offers better performance than DOS 3.3 because it employs a new CMOS 6502 instead of the Apple II's existing NMOS 6502. Advantages include faster processing, increased disk storage capacity, as well as eight new processor instructions. For a full list of features, contact Rune Software at 80 Eureka Square, Suite 214, Pacifica, CA 94044 (415) 355-4851. **Reader Service No. 115.**

Assembly language programming on the Mac for the Mac is possible with **MacASM**, a co-resident editor/macro assembler. MacASM integrates the editor, assembler, linker, resource

C Programmers B-Trees

For
\$75.00
+2.00 Postage
Source Code Included

The Softfocus B-Trees record indexing library will help you develop sophisticated application programs. With Softfocus B-Trees you get:

- high speed file handling for up to 16.7 million records per file
- customizable BDS or K & R standard C source code
- no royalties on applications programs
- support random and sequential file access
- includes example programs

Softfocus

1277 Pallatine Dr., Oakville, Ont.
Canada L6H 1Z1 (416) 844-2610



Circle no. 89 on reader service card.

Now With Windowing!
\$49.95 Basic Compiler

MTBASIC

Features:

- | | |
|--------------------|------------------|
| Multitasking | Windowing |
| Handles interrupts | Interactive |
| Fast native code | Compiles quickly |
| Floating point | No runtime fee |

MTBASIC is a true native code compiler. It runs Bytes's Sept. '81 seive in 26 seconds; interpreters take over 1400 seconds! Because MTBASIC is multitasking, it can run up to 10 Basic routines at the same time, while displaying ten separate windows. Pop-up/down menus are a snap to implement.

MTBASIC combines the best of interpreters and compilers. To the programmer, MTBASIC appears to be an extremely fast interpreter. MTBASIC compiles a typical 100 line Basic program in 1 second, yet it generates blindingly fast code. No more waiting for long compiles.

AVAILABLE for CP/M (Z-80) and PC-DOS systems.

ORDERING: Specify format when ordering. We accept Visa, MC, checks and COD. Send \$49.95 plus \$3.50 shipping and handling (\$10 overseas) to:



P.O. Box 2412 Columbia, MD 21045-1412
301/792-8096

Circle no. 88 on reader service card.

CROSS-DEVELOPMENT SOFTWARE TOOLS

C COMPILERS (with ROM support)

host | IBM/PC, target | 6809
| PDP-11,
| 6809

MACRO ASSEMBLERS

host | IBM/PC, target | 6801, 6805
| PDP-11, | 68HC11, 6809,
| 6809 | 16000, 68000

IBM/PC: TM of Int'l Business Machines.
PDP-11: TM of Digital Equipment Corp.



647 W. Virginia St.
Milwaukee, WI 53204
(414) 276-2937

Circle no. 51 on reader service card.

68000 Cross Assembler Motorola VERSAdos + Compatible

Assembler, Linker, Object and Macro Librarian. Absolute and Relocatable Code, Macros, Includes, and Conditional Assembly. Structured Programming. No limit on source file size.

Unix (C) Compatible Source
\$700

CP/M-80*	PC/DOS†	CP/M-86*
\$200	\$250	\$250

Manual: \$20
(refundable)



1329 Gregory
Wilmette, IL 60091

(312) 251-5310
after 5 p.m.

*Digital Research trademark. †IBM trademark. + Motorola trademark.

Circle no. 38 on reader service card.

New Release CP/M ↔ ISIS for PDS & MDS

ICX v.4 eXchanger now supports BOTH 8" MDS and 5-1/4" iPDS formats. Manipulation of ISIS-II files using your CP/M system was never easier.

ISE v.6 Emulator gives the CP/M-80 user access to all the ISIS-II languages and utilities.

Complete source (C and MAC asm) included with all packages

ICXMSD	\$89
ICXPDS	\$89
ISE	\$89
ICX Toolkit (all 3)	\$250

Copyrights: CPIM Digital Research, Inc.,
ISIS-II and iPDS Intel Corp.



303-327-4898 • Box C • Norwood, CO 81423

Circle no. 112 on reader service card.

microSUB:MATH

A library of Numerical Methods Subroutines for use with your FORTRAN programs.

Over sixty subroutines of:

FUNCTIONS	INTERPOLATION
INTEGRATION	LINEAR SYSTEMS
MATRICES	POLYNOMIALS
NON-LINEAR SYSTEMS	DIFFERENTIAL EQ

Versions now available for:
MS-DOS: IBM FORTRAN 77
SuperSoft FORTRAN IV
Microsoft MS-FORTRAN ver 3.2
DRI DR FORTRAN 77
CPIM-80: Microsoft F-80 FORTRAN IV
SuperSoft

LICENSE, \$250.
with SOURCE CODE, \$600.
(Manual alone, \$25.)

MATH
SUBROUTINE
LIBRARY

TRADEMARKS
Microsoft & MS: Microsoft Corp.
CPIM & DR FORTRAN 77: Digital Research Corp.
IBM: International Business Machines
SuperSoft: SuperSoft Corp.

foehn consulting, PO Box 5123, Klamath Falls, OR 97601 503/884-3023

Circle no. 39 on reader service card.

SPECIAL ONCE-ONLY OFFER

CP/M + (3.0) Source Listing
about 140 pages with
excellent comments.

Listing only
price US\$ 200.-

Listing incl. 2 Disks (8"
SS/SD) US\$ 230.-

Terms: Check or prepaid
order only.

Orders to:
CP/D oHG, West Germany
Vulkanstr. 13
4000 Duesseldorf
phone: 211-784278
tx: 858 8060

CP/M is a trademark of
DIGITAL RESEARCH

Circle no. 45 on reader service card.

TEK-MAR

HIGH-LEVEL FORTRAN GRAPHICS LIBRARY FOR THE IBM PC

Features:

- Windowing • Viewporting • Clipping
- Axis Rotation • Screen Dump & Restore
- Dump Screen Graphics to Epson

INCLUDES EXAMPLE
APPLICATION SOURCE CODE

REQUIRES:

- 320K Memory • Two DS Disk Drives
- TecMar Graphics Master Board
- MS Fortran 3.20 or higher
- Optional Plotter (Western Graphic 4636)

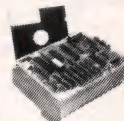
\$195

ADVANCED SYSTEMS CONSULTANTS

18653 Ventura Boulevard, Suite 351
Tarzana, California 91356
(818) 990-4942

Circle no. 3 on reader service card.

BIG DISCOUNTS ON LITTLE BOARDS™ & ACCESSORIES



- **AMPRO LITTLE BOARD™** - 64K, Z80a CPU, CTC, DART, 1 parallel port, 5 1/4 controller supports four 48tpi and/or 96tpi drives w/ CP/M 2.2 and ZCPR3 (A & T) from **\$329**
- **SYSTEM SUPPORT PKG** - Manuals, source code schematics, connectors & cables **\$99**
- **SCSI PLUS** - DMA Hard disk interface **\$99**
- **TEAC 55B DSDD** 48tpi 1/2 ht drive **\$195**
- **TEAC 55F DSDD** 96tpi 1/2 ht drive **\$239**
- **INTEGRAND** Custom two drive cabinet with 5 amp power supply & power cables **\$199**
- **TERM-MATE** - Cabinet for 2 1/2 ht + LITTLE BOARD w/ all cables & supply **\$229**
- **AMPRO SERIES 100** complete systems **\$CALL**

VISA & MASTER CHARGE, personal checks.
Please allow 2 weeks. Shipped via UPS.
Prices F.O.B. Prairie View, IL.

For additional information write or call: DISKS PLUS,
15945 West Pope Blvd., Prairie View, IL 60069
(312) 537-7888

DISKS PLUS
DIVISION OF SOLARONICS INC.

Circle no. 34 on reader service card.

compiler, and debuggers so that they can be used from the MacASM full-screen environment. Source files generated by MacASM can be edited with MacWrite, and vice versa. Introductory price is \$100.00, from Mainstay, 28611B Canwood Street, Agoura Hills, CA 91301 (818) 991-6540. **Reader Service No. 117.**

Pterodactyl Software has developed **PC BASIC**, a Basic compiler for the Apple Lisa. PC BASIC is syntax compatible with BASICA on the IBM PC. This product allows Basic programs to be quickly converted to run on the Lisa. In addition, PC BASIC provides nearly unlimited core memory space for programs and data, and allows developers to link programs to Lisa's graphics operating system, as well as Lisa's Pascal and 68000 programs. Cross-compiled and native mode versions for the Macintosh will be available soon. The price for one protected copy of the compiler is \$250.00, or \$750.00 for a runtime license that allows one to include the runtime package with applications for resale. Contact Ed Rosensweig at (415) 485-0714. **Reader Service No. 119.**

Miscellany

DriveLiner

Chandler Software has developed a portable CP/M program for verifying alignment of 8-inch floppy disk drives. A Dysan Diagnostic Diskette (see Loren Amelang's article in the December 1983 *DDJ*) is supplied with the program. Head centering, radial, and azimuth alignment tests are performed automatically on any CP/M 2.2 compatible 8-inch floppy system. The price is \$65.00 from Chandler Software, 273 West Shore Drive, Marblehead, MA 01945 (617) 631-4685. **Reader Service No. 121.**

DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service **No. 200.**



Artwork

Professional graphics capability on a personal computer for \$500.00 (IBM color graphics card) is possible with **Artwork**, an interactive graphics-design program. With the use of an input tablet or mouse, a designer can draw, modify, edit, and store complex graphic elements. Artwork includes a library of type fonts that can be scaled, rotated, italicized, or condensed. The fonts are vector-defined, not bit-mapped, so they can be changed like any other image

created by Artwork. The program also offers three-dimensional graphics manipulation capability. Images created by Artwork can be retouched by Artpaint and vice versa. The two programs are designed to complement each other. Live images from a video camera can be captured and manipulated by Artpaint. Address: West End Film Inc., 2121 Newport Place, NW, Washington, DC 20037 (202) 223-2938. **Reader Service No. 125.**



PC/AT Internal Expansion Kit

A hard disk and 1/4-inch tape drive internal upgrade kit for IBM's PC/AT converts the AT into a virtual mainframe. The kit uses IBM's hard disk controller. The Back Up and Restore Utility backs up 65 megabytes in 12 minutes. The system also allows user-configurable disk caching of up to 4

megabytes. The kit (which fits inside the AT's casing) is available in 40, 70, 140, and 280 megabytes. The internal 280 megabyte hard disk kit is priced at \$15,850.00 from Emerald Systems Corporation, 4901 Morena Boulevard, San Diego, CA 92117 (619) 270-1994. **Reader Service No. 123.**

More power to you.

Remember the magic you expected when you first purchased a PC?

It's here.

dBASE III™ is the most powerful database management system ever created for 16-bit microcomputers. It pulls every ounce of energy out of your PC and puts it to work.

On top of that, it's fast and it's easy.

You've never seen anything like it.

dBASE III can handle over a billion records per file, limited only by your computer system. You can have up to ten files open, for sophisticated applications programs.

When you have two related files, information in one can be accessed based upon data in the other.

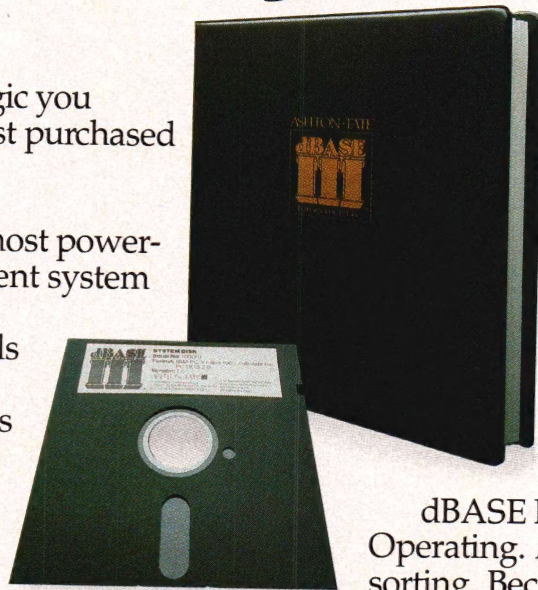
dBASE III now handles procedures, parameter passing and automatic variables. You can include up to 32 procedures in a single file. With lightning speed. Because once a file is opened, it stays open. And procedures are accessed directly.

Easier than ever.

dBASE III uses powerful yet simple commands that are the next best thing to speaking English.

If you're unsure of a command, HELP will tell you what to ask for.

If you don't know what command comes next, a command assistant does. All



you have to know is what you want it to do.

Our new tutorial/manual will have you entering and viewing data in minutes rather than reading for hours.

And to make matters easier, you get a full screen report setup for simple information access.

Faster than no time at all.

dBASE III isn't just fast. It's ultra-fast. Operating. And sorting. Even faster, is no sorting. Because dBASE III keeps your records in order, so you really don't have to sort anything. Unless you want to. Then watch out!

What about dBASE II®?

It's still the world's best database management system for 8-bit computers. And it's still the industry standard for accounting, educational, scientific, financial, business and personal applications.

Tap into our power.

For the name of your nearest authorized dBASE III dealer, contact Ashton-Tate, 10150 West Jefferson Boulevard, Culver City, CA 90230. (800) 437-4329, ext. 333. In Colorado, (303) 799-4900.

ASHTON · TATE 

© Ashton-Tate 1984. All rights reserved. dBASE III and Ashton-Tate are trademarks and dBASE II is a registered trademark of Ashton-Tate.

DeSmet C

**8086/8088
Development
Package**

\$109

FULL DEVELOPMENT PACKAGE

- Full K&R C Compiler
- Assembler, Linker & Librarian
- Full-Screen Editor
- Execution Profiler
- Complete **STDIO** Library (>120 Func)

Automatic DOS 1.X/2 X SUPPORT

BOTH 8087 AND S/W FLOATING POINT OVERLAYS

OUTSTANDING PERFORMANCE

- First and Second in AUG '83 BYTE benchmarks

SYMBOLIC DEBUGGER

\$50

- Examine & change variables by name using C expressions
- Flip between debug and display screen
- Display C source during execution
- Set multiple breakpoints by function or line number

DOS LINK SUPPORT

\$35

- Uses DOS .OBJ Format
- LINKs with DOS ASM
- Uses Lattice® naming conventions

Check: ☐ Dev. Pkg (109)
☐ Debugger (50)
☐ DOS Link Supt. (35)

SHIP TO: _____

CW ARE
CORPORATION

P.O. BOX C
Sunnyvale, CA 94087
(408) 720-9696

All orders shipped UPS surface on IBM format disks. Shipping included in price. California residents add sales tax. Canada shipping add \$5, elsewhere add \$15. Checks must be on US Bank and in US Dollars. Call 9 a.m. - 1 p.m. to CHARGE by VISA/MC/AMEX.

Circle no. 18 on reader service card.

ADVERTISER INDEX

Reader Service No.	Advertiser	Page No.	Reader Service No.	Advertiser	Page No.
1	ABCComputing	91	58	Lattice, Inc.	61
2	A\N Software Inc.	101	59	Leo Electronics, Inc.	103
3	Advanced Systems Consultants	125	60	Lifeboat Associates	64-65
4	Aim Technology	85	61	Mark Williams Co.	11
5	Amanuensis, Inc.	81	62	MicroMotion	77
6	American Planning Corp.	85	63	Micron Technology, Inc.	103
7	Ampro Computers, Inc.	99	64	Microprocessors Unlimited	103
8	Apropos Technology	77	65	Microtec Research Inc.	C-3
9	Ashton-Tate	127	66	Mitek	57
10	Avocet Systems, Inc.	33	67	NCI	115
11	B&L Computer Consultants	57	68	Opt-Tech Data Processing	103
12	BD Software, Inc.	17	69	Paragon Courseware	67
13	B.G. Micro	105	70	Phoenix Computer Products	9
14	Borland International	C-4	71	Poor Person Software	95
15	C Source	94	72	Port-A-Soft	81
16	C Systems	31	73	Procode International	73
17	C User's Group	113	74	Programmer's Connection	113
18	C Ware	128	75	The Programmer's Shop	111
19	cLINE, Inc.	7	76	QCAD Systems Inc.	99
20	Catspaw	119	77	Quic-N-Easi Products	108-109
21	Computer Friends	15	78	Raima Corp.	98
22	Computer Helper Industries	91	79	Rational Systems, Inc.	67
23	Computer Innovations	15	80	Revasco	93
24	Computer Innovations	75	81	Robotics Age	19
25	Computing	69	82	SLR Systems	91
26	Coriolis Computer	59	83	Stride Micro	3
27	Creative Solutions	49	84	Satellite Software Int'l.	C-2
28	D&W Digital	10	85	SemiDisk Systems	95
29	Data Access Corporation	21	86	Shaw American Tech.	121
30	Data Base Decisions	100	87	Simpliway Products Company	99
31	Datalight	99	88	Softaid	125
32	Dedicated Microsystems	93	89	Softfocus	125
33	Digital Research Computers	45	90	Software Horizons, Inc.	75
34	Disks Plus	125	91	Software Toolworks	83
35	Ecosoft, Inc.	69	92	Software, Inc.	69
	* Edward Ream	71	93	Solution Systems	23
36	Essential Software	71	94	Solution Systems	111
37	Faircom	85	95	Solution Systems	111
38	Farware	125	96	Solutionware	119
39	Fochm Consulting	125	97	Speedware	101
40	Fox Software Inc.	90	98	Summit Software	43
41	GGM Systems, Inc.	95	100	The Software Bottling Co.	2
42	GTEK	101	102	Thunder Software	128
43	Greenleaf Software Inc.	89	104	UNIX Systems Expo/85	53
44	Harvard Softworks	18	106	Vance Info Systems	113
45	Hoco	125	108	Visual Age	118
46	IQ Software	87	110	WL Computer Systems	57
48	Illyes Systems	30	112	Western Wares	125
49	Integral Quality	73	114	Whitesmiths Ltd.	1
50	Interface Technologies	122-123	116	Wizard Systems Software	121
51	IntrolCorporation	125	118	Workman & Associates	93
52	Key Solutions, Inc.	75	120	Zavies	81
53	Kimrich Computer Design	77	122	DDJ Back Issues	55,73
54	Korsmeyer Electronics Design	37	124	DDJ Bound Volume	104
55	Laboratory Microsystems Inc.	116	126	DDJ Reprints	71
56	Lahey Computer Systems	93	128	DDJ Subscription	67
57	Lark Software	59	132	DDJ Advertising	121

Thunder Software

- **The THUNDER C Compiler** - Operates under the APPLE Pascal 1.1 operating system. Create fast native 6502 programs to run as stand alone programs or as subroutines to Pascal programs. A major subset of the C defined by K & R. Includes a 24 page users guide, newsletters, Macro preprocessor, runs on APPLE II, II+, IIe, IIc. Source code for libraries is included. **Only \$49.95**
- **ASSYST: The Assembler System** - A complete 6502 editor/assembler and linker for APPLE DOS 3.3. Menu driven, excellent error trapping, 24 p. users guide, demo programs, source code for all programs! Great for beginners. **Only \$23.50**
- **THUNDER XREF** - A cross reference utility for APPLE Pascal 1.1. XREF generates cross references for each procedure. Source code and documentation provided. **Only \$19.95**

**Thunder Software POB 31501 Houston Tx 77231 713-728-5501
Include \$3.00 shipping. COD, VISA and MASTERCARD accepted**

Circle no. 102 on reader service card.

Software Development Tools

If:
 Manufacturer-Compatible,
 Fully-Supported Cost-
 Effective, Cross and Native
 Development tools are
 important to you!

Then:
 Let us tell you about
 MICROTEC RESEARCH's extensive
 line of field-proven software
 development tools for serious
 software developers.

Manufacturer Compatible

MICROTEC RESEARCH has been providing flexible and economical solutions for software developers since 1974. We've grown with the industry as our cross software development tools have transformed many large and small computers into powerful cross development systems for microprocessor applications. Our software tools are manufacturer compatible. Therefore, software made using manufacturers development systems may be more productively used in the MICROTEC RESEARCH cross-development environment.

Effective Differences

Beginning with product concept, through development, quality assurance, and post-sales support — **Quality, Compatibility, and Service** are the differences which set MICROTEC RESEARCH apart from the others.

Fully-Supported

MICROTEC RESEARCH's products are continually maintained and updated based upon the experience of thousands of installations worldwide. Revisions and modifications are distributed to licensees in warranty or covered by a service contract. Upgrades, which are major enhancements to a product's capabilities, are available at a significant discount. Our update/upgrade policy assures users they'll always be current with the fast moving world of microprocessor development.

Start Saving Time & Money

If you are a serious software developer, shopping for software development tools, call or write today for more information.
800-551-5554 In CA call **(408) 733-2919**

Target Microprocessors	Host Mini's	Host PC's	Software Tools
8086/80186 8096 8080/8085 8051 8048 others	DEC VAX, PDP-11 DG MV-Series Eclipse Apollo UNIX Systems others	IBM PC Data General/ONE HP 150 COMPAQ Columbia DEC Rainbow Other Compatibles	C Compilers Pascal Compilers Assemblers Simulators Linking Loaders Librarians Conversion/ Download Communications VT-100 Emulator

for Serious Software Developers

MICROTECTM
RESEARCH

3930 Freedom Circle, Suite 101, Santa Clara, CA 95054
 Mailing Address: P.O. Box 60337, Sunnyvale, CA 95088
 (408) 733-2919 Telex (ITT) 4990808

NEW from BORLAND!
TURBO TOOLBOX & TURBO TUTOR
 "TURBO is much better than the Pascal IBM sells."
 Jerry Pournelle, Byte, July 1984
 "If you have the slightest interest in Pascal—buy it."
 Bruce Webster, Softalk IBM, March 1984

BORLAND INTERNATIONAL GIFT PACK

ONLY
\$99⁹⁵
 A SAVINGS OF \$30!

What a gift for you and your friends! The extraordinary TURBO PASCAL compiler, together with the exciting new TURBO TOOLBOX and new TURBO TUTOR. All 3 manuals with disks for \$99.95.

TURBO PASCAL Version 2.0 (reg. \$49.95). The now classic program development environment still includes the FREE MICROCALC SPREAD SHEET. Commented source code on disk

• Optional 8087 support available for a small additional charge

NEW! TURBO TOOLBOX (reg. \$49.95). A set of three fundamental utilities that work in conjunction with TURBO PASCAL.

Includes:

- TURBO-ISAM FILES USING B+ TREES. Commented source code on disk
- QUIKSORT ON DISK. Commented source code on disk
- GINST (General Installation Program)

Provides those programs written in TURBO PASCAL with a terminal installation module just like TURBO'S!

• NOW INCLUDES FREE SAMPLE DATABASE

NEW! TURBO TUTOR (reg. \$29.95). Teaches step by step how to use the TURBO PASCAL development environment—an ideal introduction for basic programmers. Commented source code for all program examples on disk.

30 DAY MONEY BACK GUARANTEE
 These offers good through Feb. 1, 1985

For VISA and MASTERCARD order call toll free:

1-(800)-255-8008 1-(800)-742-1133

(Lines open 24 hrs., 7 days a week)

Dealer and Distributor inquiries welcome (408) 438-8400

CHOOSE ONE (please add \$5.00 for handling and shipping U.S. orders)

_____ All Three-Gift Pack	\$ 99.95 + 5.00 SPECIAL!
_____ All Three & 8087	139.95 + 5.00 SPECIAL!
_____ Turbo Pascal 2.0	49.95 + 5.00
_____ Turbo Toolbox	49.95 + 5.00
_____ Turbo Tutor	29.95 + 5.00
_____ Turbo 8087	89.95 + 5.00

Check _____ Money Order _____ VISA _____ MasterCard _____

Card #: _____ Exp. date: _____ Shipped UPS

My system is: 8 bit _____ 16 bit _____

Operating System: CP/M 80 _____ CP/M 86 _____ MS DOS _____ PC DOS _____

Computer: _____ Disk Format: _____

Please be sure model number & format are correct.

NAME: _____

ADDRESS: _____

CITY/STATE/ZIP: _____

TELEPHONE: _____

California residents add 6% sales tax. Outside U.S.A. add \$15.00 (if outside of U.S.A. payment must be by bank draft payable in the U.S. and in U.S. dollars). Sorry, no C.O.D. or Purchase Orders.

G11

**BORLAND
INTERNATIONAL**

4113 Scotts Valley Drive
 Scotts Valley, CA 95066
 TELEX: 172373